НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Програмна система для обліку та контролю виробництва паливних брикетів RUF»

Виконала:
студентка IV курсу, групи КП-52
Пахомова Євгенія Олександрівна _____

Керівник:
ст. викладач кафедри ПЗКС, к.т.н.,
Люшенко Л.А. _____

Консультант з нормоконтролю:
доц. каф. ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
доц. каф. ММСА ІПСА, к.ф.-м.н.,
Шубенкова І.А. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2019

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2018 р.

ЗАВДАННЯ

на дипломний проект студентці

Пахомовій Євгенії Олександрівні

1. Тема проекту «Програмна система для обліку та контролю виробництва паливних брикетів RUF», керівник проекту Люшенко Леся Анатоліївна, старший викладач, к.т.н, затверджені наказом по університету від «22» травня 2019р. №1331-С

2. Термін подання студентом проекту «19» червня 2019 р.

3. Вихідні дані до проекту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - актуальність розробки;
   - обґрунтування вибору засобів реалізації;
   - розробка системи;
   - аналіз розробленого програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - структура взаємодії представлень (креслення);
   - структура бази даних (креслення);
   - дерево проблем підприємства (плакат);
   - схема виробничої лінії (плакат).

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Вивчення літератури за тематикою проекту	14.11.2018	
2.	Розроблення та узгодження технічного завдання	28.11.2018	
3.	Розроблення структури програмного додатку	15.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	30.12.2018	
5.	Створення алгоритму роботи системи	03.02.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	20.02.2019	
7.	Програмна реалізація програмного додатку	10.03.2019	
8.	Тестування програмного додатку	17.03.2019	
9.	Підготовка матеріалів третього розділу дипломного проекту	30.03.2019	
10.	Підготовка матеріалів четвертого розділу дипломного проекту	11.04.2019	
11.	Підготовка графічної частини дипломного проекту	21.04.2019	
12.	Оформлення документації дипломного проекту	26.05.2019	

Студентка Є.О. Пахомова

Керівник проекту Л.А. Люшенко

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмної системи для обліку та контролю виробництва паливних брикетів RUF.

Програмний додаток являє собою настільний додаток, який містить 3 умовні блоки та виконує облік та контроль виробництва паливних брикетів RUF. Перший розділ відповідає за контроль виробництва та надає користувачу змогу управляти інформацією про сировину, спостерігати за візуалізованими виробничими процесами та віртуально запускати лінії по виробництву паливних брикетів, а також корегувати інформацію про кількість готової продукції на складі. Другий блок відповідає за контроль стану обладнання та надає користувачу змогу у режимі реального часу стежити за станом виробничого обладнання, додавати профілактичну роботу у графік планових профілактичних робіт обладнання, додавати у систему інформацію про відмову обладнання. Третій розділ відповідає за аналіз виробничих процесів та дозволяє переглядати звіти по роботі виробництва, завантажувати звіти по роботі виробництва, отримувати аналіз «План-факт-відхилення», отримувати від системи рекомендації по корекції виробництва.

Система розроблена для конкретного підприємства, але є масштабованою і може бути застосована на інших підприємствах такого типу.

У даному дипломному проекті розроблено: архітектуру веб-додатку, структуру бази даних, дизайн додатку, серверний модуль та модуль взаємодії зі сховищем даних.

ABSTRACT

This diploma project is devoted to the creation of a software system for the accounting and control of the production of RUF fuel briquettes.

The software application is a desktop application that contains 3 custom blocks and performs accounting and control of the production of RUF fuel briquettes. The first section is responsible for controlling production and provides the user with the ability to manage information on raw materials, observe visualized production processes and virtually launch fuel briquettes, as well as adjust information on the number of finished products in stock. The second unit is responsible for controlling the state of the equipment and provides the user with the possibility to monitor the state of the production equipment in real time, add preventive work to the schedule of planned preventive work of the equipment, to add to the system information about the failure of the equipment. The third section is responsible for the analysis of production processes and allows you to view production reports, to download reports on the work of production, to receive a "Plan-fact-deviation" analysis, to receive from the system recommendations for correction of production.

The system is designed for a particular enterprise, but is scalable and can be applied to other enterprises of this type.

In this diploma project are developed: web application architecture, database structure, application design, server module and module for interaction with the data storage.

ДП.045440-01-90 Програмна система для обліку та контролю виробництва паливних брикетів RUF. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Програмна система для	5	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Технічне завдання		
ДП.045440-03-81	Програмна система для	50	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Пояснювальна записка		
ДП.045440-04-51	Програмна система для	4	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Програмна система для	10	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Керівництво користувача		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-06-99	Програмна система для	1	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Клієнтська частина.		
	Структура взаємодії		
	представлень. Схема		
	структурна		
ДП.045440-07-99	Програмна система для	1	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	База даних системи.		
	Схема даних		
ДП.045440-08-98	Програмна система для	1	
	обліку та контролю		
	виробництва паливних		
	брикетів RUF.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ВИРОБНИЦТВА ПАЛИВНИХ БРИКЕТІВ RUF

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Є.О. Пахомова

2018

ЗМІСТ

1. Найменування та галузь застосування 3
2. Галузь розроблення 3
3. Призначення розробки 3
4. Галузь розроблення 3
5. Вимоги до проектної документації 4
6. Етапи проектування 5
7. Порядок тестування розробки 5

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Програмна система для обліку та контролю виробництва паливних брикетів RUF.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання на виробництві брикетів RUF для обліку та контролю стану виробництва, контролю стану обладнання у цеху, аналізу виробництва.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Програмний додаток повинен забезпечувати такі основні функції:

- перегляд інформації про кількість сировини у бункері;
- корегування інформації про кількість сировини у бункері;
- ввід даних про кількість сировини, з якої будуть вироблені паливні брикети RUF;
- віртуальний запуск лінії по виробництву паливних брикетів;
- візуалізація виробничих процесів у системі;
- перегляд інформації про кількість готової продукції на складі;
- коригування інформації про кількість готової продукції на складі;

- перегляд звітів по роботі виробництва за обраний період;
- завантаження звітів по роботі виробництва за обраний період;
- отримання аналізу «План-факт-відхилення» за обраний період часу;
- отримування від системи рекомендації по корекції виробництва для його оптимізації;
- перегляд інформації про стан обладнання на виробництві;
- додавання профілактичних робіт у графік планових профілактичних робіт обладнання;
- додавання у систему інформації про відмову обладнання.

Розробку виконати з використанням веб-фреймворку ReactJS мовою JavaScript.

Додаткові вимоги:

- інтуїтивно-зрозумілий інтерфейс;
- кросплатформеність додатку.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

- пояснювальна записка;
- програма та методика тестування;
- керівництво користувача;
- креслення:
  - «Клієнтська частина. Структура взаємодії представлень»;
  - «База даних системи».

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи..... 14.11.2018

Розроблення та узгодження технічного завдання 28.11.2018

Розроблення структури програмного додатку..... 15.12.2018

Розроблення дизайну сторінок 03.02.2019

Програмна реалізація програмного додатку 17.03.2019

Тестування програмного додатку..... 03.04.2019

Підготовка матеріалів текстової частини проекту 28.04.2019

Підготовка матеріалів графічної частини проекту..... 28.04.2019

Оформлення технічної документації проекту..... 25.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___"_____2019 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ВИРОБНИЦТВА ПАЛИВНИХ БРИКЕТІВ RUF

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проекту:

_____Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____Є.О. Пахомова

2019

ЗМІСТ

ВСТУП.....4

МЕТА ДИПЛОМНОГО ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ.....5

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 6

1. АКТУАЛЬНІСТЬ РОЗРОБКИ 7

1.1. Поточний стан технології в Україні 7

1.2. Огляд існуючих програмних рішень 8

1.3. Апаратні засоби та обладнання 11

1.4. Вимоги до програмного забезпечення 12

1.5. Обгрунтування актуальності розробки 13

1.6. Висновки 14

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ.....15

2.1. Обґрунтування вибору мови програмування 15

2.2. Обґрунтування вибору системи керування базами даних 23

2.3. Висновки 28

3. РОЗРОБКА СИСТЕМИ 30

3.1. Загальний опис системи.....30

3.2. Функціональні вимоги до системи 33

3.3. Архітектура системи 34

3.4. Висновки 38

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....39

4.1. Аналіз реалізації вимог.....39

4.2. Тестування системи.....40

4.3. Порівняння розробки з існуючими аналогами 42
4.4. Рекомендації щодо подальшого вдосконалення 43
4.5. Висновки 44
ВИСНОВКИ.....46
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ.....48
ДОДАТКИ.....51

ВСТУП

Технічний прогрес ніколи не стоїть на місці. Технологічні новинки та нові програмні розробки покликані покращити життя людей. Однією з найбільш проблемних галузей, що потребують розвитку, є ручне управління виробництв малих та великих масштабів. Людський ресурс коштує дорого та потребує регулярної оплати. Натомість, достатньо зробити одне велике вкладення грошей у якісне програмне забезпечення, щоб значно зменшити витрати на людський ресурс назавжди.

Виробництво паливних брикетів RUF є однією з найприбутковіших галузей паливної промисловості в Україні і має достатньо складний виробничий процес. Саме тому створення автоматизованої системи контролю та обліку виробництва паливних брикетів RUF є важливою задачею для успішного функціонування підприємства. Також розроблена система є масштабованою та може застосовуватись для інших малих та великих виробництв. Таким чином також вдасться досягти зменшення браку на виробництві, адже майже зникне так званий «людський фактор».

Даний дипломний проект присвячено розробленню системи, що вирішує проблеми підприємства, що займається виробництвом паливних брикетів RUF. Розробка виконана у вигляді настільного кросплатформеного додатку, який зможуть використовувати як оператори цеху, так і керівники підприємства. Керівництву не потрібно спускатися в цех для контролю виробництва. Потрібен лише комп’ютер з установленим застосунком, який надасть таку можливість. А використання застосунку на комп’ютері у цеху дозволить операторам оптимізувати складний виробничий процес.

МЕТА ДИПЛОМНОГО ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ

Метою даного дипломного проекту є розроблення програмної системи, яка виконувала б облік та контроль виробництва паливних брикетів RUF.

Відповідно до мети написання дипломного проекту та розроблення програмного застосунку були поставлені і розв’язані такі задачі:

- проведення огляду та аналізу існуючих аналогів, а саме програмних рішень та відомих технологій для з’ясування їх сильних та слабких місць, на основі яких відбулося складання вимог до розроблюваної системи;
- вибір засобів реалізації програмного застосунку для дипломного проекту, зокрема таких як мови програмування, середовища для розробки веб-застосунків, системи управління базою даних та архітектурного шаблону;
- розроблення структурно-алгоритмічної організації програмного забезпечення;
- реалізація й тестування програмного забезпечення та проведення аналізу отриманого результату.

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Android – операційна система і платформа для мобільних телефонів та планшетних комп’ютерів, створена компанією Google на базі ядра Linux;

API – Application Programming Interface (прикладний програмний інтерфейс);

ASP – Active Server Pages (активні серверні сторінки);

CSS – Cascading Style Sheets (спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних);

Framework – інфраструктура програмних рішень, що полегшує розробку складних систем;

HTML – HyperText Markup Language (мова розмітки гіпертекстових документів);

HTTP – Hyper Text Transfer Protocol (протокол передачі даних, що використовується в комп’ютерних мережах);

JS – JavaScript (динамічна, об’єктно-орієнтована мова програмування);

SQL – Structured Query Language (декларативна мова програмування для взаємодії користувача з базами даних);

URL – Uniform Resource Locator (стандартизована адреса певного ресурсу);

XML – Extensible Markup Language (розширювана мова розмітки);

Аутентифікація – процедура встановлення належності користувачеві інформації в системі пред'явленого ним ідентифікатора;

БД – база даних;

СКБД – система керування базами даних.

1. АКТУАЛЬНІСТЬ РОЗРОБКИ

1.1. Поточний стан технології в Україні

У наш час найбільшою проблемою розвитку промисловості в Україні є ручне управління виробництвом. На багатьох виробництвах, в тому числі на виробництві паливних брикетів RUF найменшу задачу, виконує людина [1]. Незалежно від того, наскільки ця задача примітивна та рутинна. Це несе за собою такі проблеми, як значні витрати на людський ресурс, а також неоптимальну роботу виробництва, управління яким відбувається вручну.

Якщо розглянути поточний стан технологій управління виробництвом паливних брикетів RUF в Україні, то можна побачити багато проблем. Першою з них є відсутність планування виробництва. Все відбувається сумбурно, без чітких термінів та об’ємів поставок.

Другою проблемою є відсутність інформації про реальне виробництво. Інформація або зберігається у неструктурованому вигляді на аркушах, або відсутня взагалі. Це тягне за собою такі наслідки як відсутність бази даних показників виробничих процесів, відсутність даних про виробничі запаси сировини, відсутність інформації про готову продукцію, відсутність системи обрахунку показників для прийняття рішень по виробничим процесам [2].

Третьою проблемою є відсутність аналізу виробничих показників. Навіть якщо дані про показники якимось чином були збережені (що відбувається вкрай рідко), їх аналіз не відбувається ніколи. Це заважає підприємству розвиватися та вдосконалювати свою роботу. Адже повністю відсутні звіти по виробництву, тому відсутній класичний аналіз «план-факт-відхилення», а також відсутні рекомендовані коригувальні дії.

Останньою, але не за значенням, проблемою є ручний контролю стану обладнання [3]. Все контролюють люди. Цей факт призводить до раптових поломок та неоптимального використання виробничих

потужностей. Причиною цього є відсутність інформації про стан обладнання, відсутність графіку планових профілактичних робіт обладнання, відсутність інформації про відмови та помилки у роботі обладнання.

Таким чином поточний стан технології автоматизації управління виробництвом паливних брикетів RUF в Україні далеко не найкращий. Існує багато проблем, які сильно впливають на роботу підприємства в цілому. Єдиним виходом з ситуації є створення програмного забезпечення, яке б виконувало облік та контроль виробництва паливних брикетів RUF та назавжди витіснило ручне управління з підприємства.

1.2. Огляд існуючих програмних рішень

Бітрікс24

Бітрікс24 – програмний застосунок, який дозволяє керувати роботою підприємства. Являє собою потужну ERP-систему. Включає 5 компонентів: CRM, Завдання і проекти, Контакт-центр, Сайти та магазини, Компанія, див. рис. 1. Охоплює інтеграцію виробництва і операцій, управління трудовими ресурсами, фінансовий менеджмент і управління активами. Є можливість як роботи онлайн на сайті самого сервісу, так і придбання «Коробкової версії», яка має відкритий код та встановлюється на сервер підприємства, що обслуговується. Контролює задачі підприємства, організовує внутрішній зв’язок. Аналізує отримані дані та робить розгорнуті звіти про роботу підприємства. Може бути шляхом довгого налаштування придатний для вирішення усіх вищеописаних проблем. Але також має багато додаткового функціоналу, який не буде використаним.

Бітрікс24 – це оптимальний варіант програмного забезпечення для підприємств широкого профіля. Програмний застосунок може взяти під контроль управління усім великим підприємством. Але це не оптимальний варіант для невеликих виробництв, таких як виробництво паливних

брикетів RUF. Адже налаштування усіх параметрів цієї потужної системи під потреби малого підприємства вимагатиме роботи цілої команди програмістів та потягне за собою витрати у розмірі приблизно 20 тисяч доларів. При цьому велика частина функціоналу системи так і не буде задіяна.

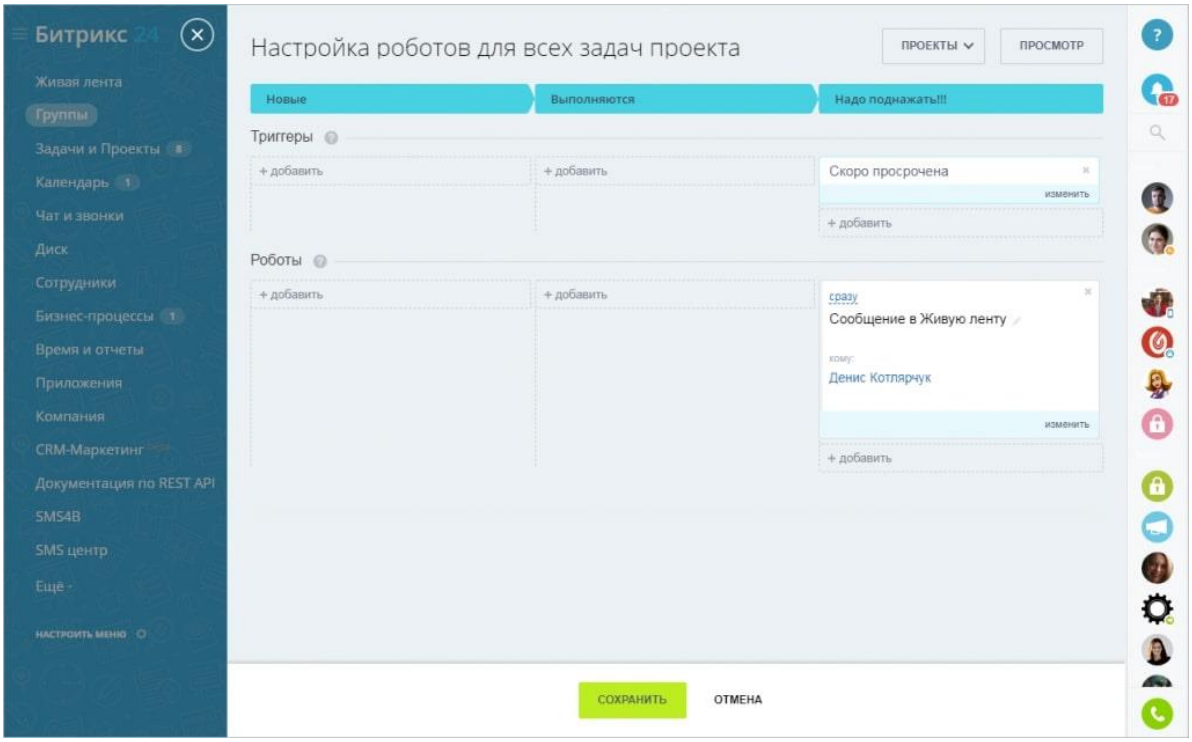


Рис. 1. Веб-сторінка сайту Бітрікс24 для керування задачами

Саме через описані вище чинники системи Бітрікс24 не підходить для використання на малому підприємстві для обліку та контролю виробництва паливних брикетів RUF. Оптимальним варіантом буде спеціально розроблене програмне забезпечення, яке б вирішувало тільки конкретні проблеми вузького спектру підприємств та не потребувало великої кількості додаткових налаштувань.

1С:Підприємство

1С: Підприємство – програмний продукт компанії «1С», призначений для управління діяльністю на підприємстві. Це потужна ERP система, яка покриває усі області роботи великих підприємств. Програмне

забезпечення розроблене для контролю бухгалтерського та управлінського обліків (включаючи нарахування зарплати і управління кадрами), економічної та організаційної діяльності підприємства. Це ПЗ є спеціалізованим інтегрованим пакетом прикладного програмного забезпечення, що забезпечує загальну модель даних і процесів для всіх сфер діяльності підприємства з основною функцією бухгалтерського та податкового обліку. Нова версія системи також надає можливість роботи з Конфігуратором, що дозволяє на програмному рівні керувати роботою системи та корегувати її можливості залежно від потреб підприємства. Приклад роботи з Конфігуратором представлений на рис. 2.

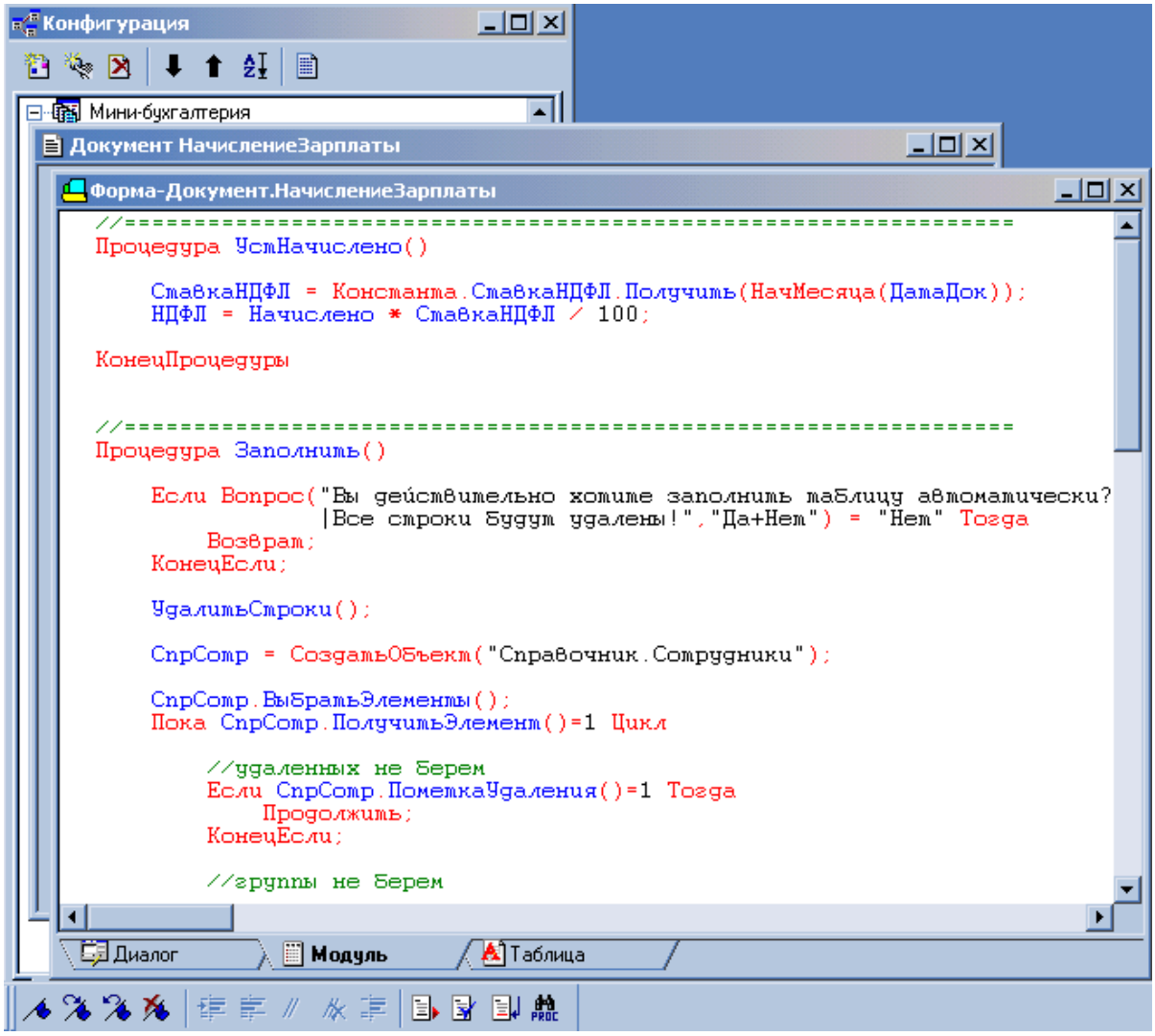


Рис. 2. Приклад роботи з 1С. Конфігуратор

Архітектурно програмний застосунок являє собою програмну оболонку над базою даних. Платформа має свою внутрішню мову програмування, яка забезпечую доступ до даних а також взаємодію з іншими програмами. Може взаємодіяти з Microsoft Windows, Mac OS X, а також Linux.

Вищеописане програмне забезпечення є чудовим варіантом для великих підприємств, що мають необхідну кількість для налаштування системи та навчання співробітників роботі з нею. Адже ПЗ «1С: Підприємство» може бути кероване тільки висококваліфікованими спеціалістами. У випадку невеликого підприємства, що займається виробництвом паливних брикетів RUF, потреб у такій великій та потужній ERP-системі немає. Як і немає ресурсів для навчання співробітників роботі з системою, або найму висококваліфікованих спеціалістів. Не дивлячись на те, що з допомогою команди програмістів система може бути налаштована для роботи з виробництвом паливних брикетів RUF, це не є оптимальним варіантом. Адже велика частина системи буде не задіяна. Тому найкращим варіантом для вирішення усіх проблем ручного управління виробництвом буде створення спеціалізованого програмного забезпечення для обліку та контролю виробництва паливних брикетів RUF.

1.3. Апаратні засоби та обладнання

Основною задачею дипломної роботи є розроблення автоматизованої системи для контролю та обліку виробництва паливних брикетів RUF. У той самий час система має бути достатньо гнучкою та масштабованою. Має бути можливість застосування системи (після деяких доробок, звичайно) для контролю та обліку виробництв малих і великих підприємств інших сфер промисловості. Але у рамках цієї дипломної роботи будуть розглянуті апаратні

засоби та обладнання, які використовуються саме для виробництва паливних брикетів RUF.

Цех, що виробляє паливні брикети RUF, обладнаний лінією по виробництву брикетів та комп'ютером.

У цеху підприємства, лінія по виробництву паливних брикетів RUF складається з:

- різчика;
- бункера;
- печі;
- вологоміру;
- пресу (послідовно розміщені шнек та матриця);
- вагів.

Усі елементи розміщені послідовно та з'єднані конвеєром [4]. Виробник та програмний інтерфейс обладнання не має значення, адже всі маніпуляції будуть проводитися черговими по зміні інженерами, які, у свою чергу, будуть керуватися даними з розробленого програмного забезпечення.

Комп'ютер, що знаходиться в цеху, має бути достатньо потужним для цілодобової роботи та обробки потому інформації [5]. Об'єм оперативної пам'яті – не менше 8ГБ. Операційна система не матиме значення, адже однією в вимог до розроблюваного програмного забезпечення є кросплатформеність. Може використовуватись як стаціонарний ПК, так і ноутбук.

1.4. Вимоги до програмного забезпечення

Вимоги до інтерфейсів:

- функціональність (відповідність задачам користувача);
- зрозумілий користувачу інтерфейс;

- адаптивність до вирішення поставленої задачі;
- простота експлуатації, максимальний час навчання персоналу – 3 робочі дні.

Вимоги до апаратної частини:

- ноутбук або ПК;
- операційна система Windows, Mac OS або Linux;
- підключення до мережі Інтернет через Wi-Fi модуль.

Вимоги до масштабування:

- стійкість системи до масштабування.

Вимоги до безпеки:

- безпека зберігання даних у базі;
- стійкість до хакерських атак.

Вимоги до продуктивності:

- швидкодія системи, максимальний час відгуку – 1 секунда;
- збереження у базу всіх даних, введених в систему;
- максимальний час оновлення екрана – 3 секунди;
- самостійне відновлення роботи системи після помилки.

1.5. Обгрунтування актуальності розробки

Після детального дослідження теми стає очевидно, що проблема ручного управління виробництвом є значною та потребує найскорішого вирішення.

Проблема ручного управління є глобальною та тягне за собою такі наслідки, як відсутність планування виробництва, відсутність даних про реальний стан виробництва, відсутність аналізу виробничих показників та звітів по роботі, ручний контроль обладнання.

Усі ці проблеми не дають підприємству розвиватися, а також значно знижують щомісячний прибуток. Саме тому розробка програмного забезпечення, яке вирішило б вищеописані проблеми виробництва

паливних брикетів RUF є дуже актуальною задачею.

Створення автоматизованої системи обліку та контролю виробництва паливних брикетів RUF дозволить підприємству домогтися сталого розвитку, звільнити та перерозподілити ресурси, покращити якість вихідної продукції та збільшити прибуток. Концентрація усіх цих факторів можлива тільки завдяки розробці вищеописаного програмного забезпечення. Розробку можна вважати цілком актуальною, адже вона є єдиним шляхом до успіху підприємства, що займається виробництвом паливних брикетів RUF.

1.6. Висновки

Як бачимо, актуальність створення системи обліку та контролю виробництва паливних брикетів RUF є значною, а огляд існуючих рішень і порівняння з ними дають розуміння того, що ці рішення не можуть оптимально вирішити описані вище проблеми українських підприємств. Одним з найбільших недоліків, які не дають українським підприємствам по виробництву паливних брикетів RUF використовувати вже існуючі рішення, є їх потужність та складність у адаптації до виробництва малого підприємства вузького профіля. Ці фактори роблять вирішення проблем підприємства за допомогою вже існуючих програмних рішень неоптимальним.

Аналізуючи даний недолік, необхідно створити Автоматизовану систему обліку та контролю виробництва паливних брикетів RUF та інтегрувати її в українське виробництво, повність замінивши нею застаріле ручне управління. Це дасть змогу вирішити основні проблеми підприємства без постійних значних грошових витрат та дасть підприємству можливість сталого розвитку.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору мови програмування

Огляд мови програмування JavaScript

JavaScript (часто скорочено як JS) – мова програмування високого рівня, що відповідає специфікації ECMAScript. JavaScript має синтаксис фігурних дужок [6], динамічну типізацію, об'єктно-орієнтовану орієнтацію на основі прототипу та функції першого класу.

Поряд з HTML і CSS, JavaScript є однією з основних технологій мережі Інтернет (World Wide Web). JavaScript дозволяє інтерактивні веб-сторінки і є важливою частиною веб-додатків. Переважна більшість веб-сайтів використовують цю мову, а основні веб-браузери мають спеціальний JavaScript-механізм для її виконання.

Як мультипарадигмова мова, JavaScript підтримує функціональні та імперативні (включаючи об'єктно-орієнтовані та прототипні) стилі програмування. Вона має API для роботи з текстом, масивами, датами, регулярними виразами та DOM, але сама мова не включає жодного вводу-виводу, наприклад мереж, сховищ або графічних засобів. Вона покладається на хост-середовище, в яке вона вбудована, щоб забезпечити ці функції.

Спочатку були реалізовані тільки клієнтські веб-браузери. Тепер двигуни JavaScript вбудовані в багато інших типів програмного забезпечення, включаючи веб-сервери та бази даних, а також у програми, не пов’язані з мережею Інтернет, такі як текстові процесори та програмне забезпечення PDF. А також ті, які під час роботи середовища роблять JavaScript доступним для написання мобільних і настільних додатків, включаючи віджети для робочого столу [7]. Саме ця властивість і може бути використна під час розробки вищеописаного програмного забезпечення.

Терміни Vanilla JavaScript і Vanilla JS посилаються на JavaScript, не

розширений жодними рамками або додатковими бібліотеками. Скрипти, написані на Vanilla JS, є звичайним JavaScript кодом.

Незважаючи на наявність подібності між JavaScript і Java, включаючи назву мови, синтаксис і відповідні стандартні бібліотеки, ці дві мови відрізняються на всіх етапах розробки. На JavaScript впливали такі мови програмування, як Self і Scheme.

Основні властивості мови JavaScript:

- JavaScript є легкою, інтерпретованою мовою програмування;
- призначена для створення мережевих орієнтованих додатків;
- доповнює та інтегрується з Java;
- доповнює та інтегрується з HTML;
- відкрита і крос-платформнена.

На стороні клієнта JavaScript є найпоширенішою мовою. Скрипт повинен бути включений в HTML документ або на нього посилатися, щоб код був інтерпретований браузером. Це означає, що веб-сторінка не повинна бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером і динамічно створюють вміст HTML.

Механізм JavaScript на стороні клієнта надає багато переваг перед традиційними CGI-серверними скриптами. Наприклад, можна скористатися JavaScript, щоб перевірити, чи введе користувач дійсну адресу електронної пошти у полі форми. Код JavaScript виконується, коли користувач подає форму, і тільки якщо всі записи дійсні, вони будуть подані на веб-сервер.

JavaScript може використовуватися для уловлювання ініційованих користувачем подій, таких як кліки кнопок, навігація посилань та інші дії, які користувач ініціює явно або неявно.

Переваги мови JavaScript:

- менша взаємодія з сервером. Можна перевірити вхід користувача перед відправкою сторінки на сервер. Це економить трафік сервера, що означає менше навантаження на ваш сервер;
- негайний зворотний зв'язок з відвідувачами. Не потрібно чекати, поки сторінка перезавантажиться, щоб побачити, чи вони забули щось ввести;
- збільшена інтерактивність. Можна створювати інтерфейси, які реагують, коли користувач зависає над ними за допомогою миші або активує їх за допомогою клавіатури;
- сучасність інтерфейсів. Можна використовувати JavaScript, щоб включити такі елементи, як компоненти перетягування і повзунки, щоб надати відвідувачам свого сайту багатий інтерфейс.

Недоліки мови JavaScript:

- не можна розглядати JavaScript як незалежну мову програмування;
- на стороні клієнта JavaScript не дозволяє читати або писати файли, це відбувається з міркувань безпеки;
- JavaScript не можна використовувати для мережевих програм;
- у JavaScript відсутні багатопоточні або багатопроцесорні можливості.

Огляд мови програмування Java

Java – це мова програмування загального призначення, що базується на класах, об'єктно-орієнтована, і розроблена спеціально для того, щоб мати якомога менше залежностей реалізації. Призначена для того, щоб розробники програм могли запустити написаний код майже у будь-якому середовищі. Це означає, що скомпільований код Java може працювати на всіх платформах, які підтримують Java без необхідності перекомпіляції [8]. Програми Java зазвичай компілюються до байт-коду, який може працювати на будь-якій віртуальній машині Java (JVM) незалежно від базової архітектури комп'ютера.

Синтаксис Java схожий на C і C ++, але він має менше можливостей

низького рівня, ніж будь-який з них. Станом на 2018 рік, Java, відповідно до GitHub, була однією з найпопулярніших використовуваних мов програмування, зокрема, для веб-додатків клієнт-серверної архітектури, з 9 мільйонами розробників.

Java була розроблена Джеймсом Гослінoм у Sun Microsystems (який з тих пір був придбаний Oracle) і випущена в 1995 році як основний компонент платформи Java Sun Microsystems. Оригінальні та довідкові реалізації Java-компіляторів, віртуальних машин і бібліотек класів спочатку були випущені Sun під власними ліцензіями. Станом на травень 2007 року, відповідно до специфікацій Java Community Process, Sun повторно ліцензувала більшість своїх технологій Java за ліцензією GNU General Public License. Тим часом інші розробили альтернативні реалізації цих технологій Sun, такі як компілятор GNU для Java (компілятор bytecode), GNU Classpath (стандартні бібліотеки) і IcedTea-Web (браузерний додаток для аплетів).

Остання версія є Java SE 12, випущена в березні 2019 року. Оскільки Java 9 більше не підтримується, Oracle радить своїм користувачам негайно перейти до Java 12 для комерційного використання, у січні 2019 року. Java 8 буде підтримуватися публічними оновленнями для особистого користування принаймні до грудня 2020 року з питань безпеки. Розширена підтримка Oracle для Java 6 завершилася в грудні 2018 року.

Переваги мови Java:

- простота: Java була розроблена, щоб бути простою у використанні і тому код легше писати, компілювати, налагоджувати та вивчати саму мову, порівняно з іншими мовами програмування. Причина, чому Java набагато простіше, ніж C++, полягає в тому, що Java використовує автоматичне виділення пам'яті і збирання «сміття»;
- об'єктно-орієнтованість: Java є об'єктно-орієнтованою, тому програмування на Java зосереджено на створенні об'єктів,

маніпулюванні об'єктами і спільній роботі. Це дозволяє створювати модульні програми і багаторазовий код;

- незалежність від платформи. Однією з найважливіших переваг Java є її здатність легко переміщатися з однієї комп'ютерної системи в іншу. Можливість запускати одну і ту ж програму на багатьох різних системах має вирішальне значення для програмного забезпечення World Wide Web, і Java досягає успіху, будучи незалежною від платформи як на початковому, так і на двійковому рівнях;
- дистрибутивність. Розподілені обчислення включають кілька комп'ютерів у мережі, що працюють разом. Java розроблена таким чином, щоб спростити розподілені обчислення за допомогою мережевих можливостей, які за своєю природою інтегровані в неї;
- інтерпретовність. Для запуску Java-програм потрібний інтерпретатор. Програми компілюються в код віртуальної машини Java, який називається байт-кодом. Байт-код є незалежним від машини і може працювати на будь-якій машині, яка має інтерпретатор Java. За допомогою Java програму потрібно компілювати тільки один раз, а байт-код, створений компілятором Java, може працювати на будь-якій платформі;
- безпека. Java є однією з перших мов програмування, яка розглядає безпеку як частину свого дизайну. Мова Java, компілятор, інтерпретатор і середовище виконання були розроблені з урахуванням безпеки;
- надійність. Java багато уваги приділяє ранньої перевірці можливих помилок, оскільки компілятори Java здатні виявити багато проблем, які з'являться вже під час виконання коду на інших мовах;
- багатопоточність. Мова надає можливість для виконувати декілька завдань одночасно в межах програми. У Java багатопоточні

програми плавно інтегровані в неї, в той час як в інших мовах потрібно викликати специфічні для операційної системи процедури, щоб увімкнути багатопоточність. Багатопоточність – це необхідність у візуальному та мережевому програмуванні.

Недоліки мови Java:

- продуктивність. Java може бути сприйнята як значно повільніша та більш споживаюча пам'ять, ніж нативно скомпільовані мови, наприклад C або C ++;
- зовнішній вигляд. За промовчанням графічні програми, написані на Java за допомогою інструментарію Swing, дуже відрізняються від нативних програм. Можна створити інший вигляд і сприйняття за допомогою підключення системи зовнішнього вигляду Swing;
- єдина парадигма: Java є переважно мовою єдиної парадигми. Однак, з додаванням статичного імпорту в Java 5.0, процедурна парадигма краще пристосована, ніж у попередніх версіях Java.

Огляд мови програмування C#

C# – це мова багатопрофільної парадигми програмного забезпечення загального призначення, що охоплює сильно типізовані, лексично обговорювані, імперативні, декларативні, функціональні, загальні, об'єктно-орієнтовані (на основі класів) та компонентно-орієнтовані дисципліни. Мова була розроблена в 2000 році корпорацією Майкрософт в рамках її ініціативи .NET і пізніше затверджена в якості стандарту Ecma (ECMA-334) і ISO (ISO/ IEC 23270: 2018). C# є однією з мов програмування, призначених для спільної мовної інфраструктури.

Мова C# була розроблений Андерсом Хейлсбергом, його команду розробників в даний час очолює Мадс Торгерсен. Найновішою версією є C# 7.3, яка була випущена в 2018 році разом з Visual Studio 2017 версією 15.7.2.

Синтаксис C# є дуже виразним, але він також простий і легкий в освоєнні. Синтаксис фігурної дужки C# буде миттєво впізнаваний всіма,

хто знайомий з C, C++ або Java. Розробники, які знають будь-яку з цих мов, зазвичай здатні продуктивно працювати з C# після дуже короткого часу освоєння. Синтаксис C# спрощує багато складнощів C++ і надає потужні функції, такі як типи значень з нулями, перерахування, делегати, лямбда-вирази і прямий доступ до пам'яті, які не доступні в Java. C# підтримує загальні методи та типи, які забезпечують підвищену безпеку та продуктивність. А також ітератори, які дозволяють реалізаторам класів колекцій визначати поведінку користувальницьких ітерацій, які є простими у використанні клієнтським кодом. Вирази з інтегрованими мовними запитами (LINQ) роблять сильно типізований запит першокласною мовою.

Переваги мови C#:

- це чисто об'єктно-орієнтована мова, що дозволяє створювати модульні доглядові програми і багаторазові коди. Це одна з найбільших переваг C # над C ++;
- мова має дуже ефективну систему для видалення всього сміття, присутнього в системі. C# не створює безлад у системі, і система не зависає під час виконання;
- мова має велику перевагу сильної резервної копії пам'яті. У C# немає проблем з витоком пам'яті та інших подібних проблем, як це відбувається у випадку з мовою C++. У цьому випадку C# має дуже чітку різницю від інших мов;
- багаті бібліотеки класів полегшують реалізацію багатьох функцій. C# впливає на більшість програмістів світу і має історію в світі програмування;
- програми, написані на .NET, матимуть кращу інтеграцію та взаємодію з іншими технологіями NET. Фактично C# працює на CLR, що дозволяє легко інтегруватися з компонентами, написаними іншими мовами (зокрема, CLR-сумісними мовами);

- формалізована концепція get-set методів, код стає більш прозорим. Також у C# не потрібно турбуватися про файли заголовків;
- можливість придбати підтримку від Microsoft у C# (.NET framework) на відміну від Java, де підтримку надає тільки спільнота розробників.

Недоліки мови C#:

- C# виконується повільніше. Це дещо згладжується при використанні WPF, хоча в даний час запуск програми WPF все ще трохи повільний [9]. Однак, після запуску програми, ефекти анімації зглажуються;
- C# є менш гнучким, ніж C++. C# значно залежить від .NET framework. Все, що не знайдено в .NET framework, буде майже неможливо реалізувати;
- Мова C# підтримується тільки операційною системою Windows, для якої вона і була розроблена.

Порівняльний аналіз мов програмування з точки зору вимог до системи

З точки зору вимог до системи, мова програмування має вирішувати такі задачі: кросплатформеність [16], гнучкий інтерфейс, стійкість до масштабування, стійкість до хакерських атак, швидкодія.

Мова програмування Java не може забезпечити такі параметри як кросплатформеність та гнучкий інтерфейс.

Мова програмування C# не може забезпечити такі параметри як кросплатформеність та швидкодія (порівняно з іншими мовами).

Лише мова програмування JavaScript підходить по всім параметрам та може вирішити поставлені задачі. Кросплатформеність забезпечується за допомогою використання фреймворку Electron [13], який легко приєднується до програми [11]. Гнучкий інтерфейс забезпечується за допомогою таких інструментів як мови HTML та CSS. Стійкість до хакерський атак, стійкість до масштабування та швидкодія притаманні

самій мові JavaScript як інструменту розробки.

2.2. Обґрунтування вибору системи керування базами даних

Огляд СКБД MySQL

MySQL – система керування реляційними базами даних [17] з відкритим вихідним кодом (RDBMS). Її назва являє собою поєднання "My", ім'я дочки співзасновника Michael Widenius, і "SQL", абревіатура Structured Query Language.

MySQL є безкоштовним і відкритим програмним забезпеченням згідно з умовами GNU General Public License, а також доступна під різними власними ліцензіями. MySQL була власником і спонсором шведської компанії MySQL AB, яка була куплена компанією Sun Microsystems (нині Oracle Corporation). У 2010 році, коли компанія Oracle придбала Sun, компанія Widenius розігнала проект з відкритим вихідним кодом MySQL для створення MariaDB.

MySQL є складовою частиною стеку програмного забезпечення LAMP (та інших), що є абревіатурою для Linux, Apache, MySQL, Perl / PHP / Python. MySQL використовується багатьма веб-додатками, що керуються базами даних, включаючи Drupal, Joomla, phpBB і WordPress. MySQL також використовується багатьма популярними веб-сайтами, включаючи Facebook, Twitter, Flickr, і YouTube.

MySQL написана на C та C ++. Її SQL-парсер написана на yacc, але вона використовує домашній лексичний аналізатор. MySQL працює на багатьох системних платформах, включаючи AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, MacOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos і Tru64. Також існує порт MySQL до OpenVMS.

Серверне програмне забезпечення MySQL і клієнтські бібліотеки використовують розподіл з двома ліцензіями. Вони пропонуються за

версією GPL 2 або ліцензією на власність.

Підтримку можна отримати з офіційного посібника. Безкоштовна підтримка додатково доступна в різних каналах і форумах IRC. Oracle пропонує платну підтримку через продукти MySQL Enterprise. Вони відрізняються за обсягом послуг і в ціні. Крім того, існує ряд організацій третіх сторін, які надають підтримку та послуги, включаючи MariaDB та Percona.

MySQL отримала позитивні відгуки. СКБД також був протестована і визначена як "швидка, стабільний і справжня багатокористувацька, багаточна SQL-серверна база даних" [18].

Переваги СКБД MySQL:

- MySQL дуже проста у встановленні, і завдяки базі сторонніх інструментів, які можуть бути додані до бази даних, налаштування реалізації є відносно простим завданням. Крім того, це також легка база даних, з якою можна працювати. Поки є розуміння мови, не відбудеться стикання з надто великими проблемами;
- існує велика і процвітаюча спільнота розробників і ентузіастів, до яких можна звернутися за допомогою. Це пояснюється значною мірою популярністю рішення, кінцевим результатом якого є нестача фахівців;
- СКБД може вважатися опцією бази даних з відкритим вихідним кодом, оскільки код все ще доступний безкоштовно онлайн;
- реалізація MySQL може варіюватися в ціні від безкоштовного до $10,000 або більше. У будь-якому випадку це значно дешевше, ніж більшість інших опцій бази даних на ринку (за винятком конкурентів з відкритим кодом MySQL);
- хоча популярність MySQL дещо зменшилася в останні роки, вона залишається однією з найбільш використовуваних систем баз даних у світі. СКБД сумісна практично з усіма операційними системами і є більш-менш промисловим стандартом.

Недоліки СКБД MySQL:

- MySQL має тенденцію бути дещо менш надійною, ніж її конкуренти. Ці питання стабільності пов'язані з тим, як вона виконує певні функції (наприклад, посилання, транзакції та аудит);
- хоча MySQL обладнана практично безмежним обсягом даних, вона має тривожну тенденцію до завершення роботи, якщо буде змушена мати справу з занадто великою кількістю операцій в даний момент часу. Це відносно низьке масштабування продуктивності означає, що будь-хто з високими рівнями паралелізму, ймовірно, має шукати альтернативу;
- компанія не приймає доробки, створені спільнотою розробників, і не намагається запропонувати користувачам будь-яку інструкцію для розробки з MySQL. Для розробників дійсно немає можливості обговорювати систему керування базами даних з Oracle;
- СКБД не є повністю сумісною з SQL та має тенденцію бути обмеженою в областях, включаючи зберігання даних, відмовостійкість та діагностику продуктивності. Розробники можуть виявити цю відсутність функціональності розчаровуючою, особливо якщо вони звикли до більш повнофункціональної альтернативи;
- MySQL має меншу функціональність, ніж багато інших СКБД на ринку. Деякі функції, такі як пошук тексту та відповідність ACID, залежать не від основного двигуна, а від додатків. Хоча це правда, що існує безліч добре створених додатків для MySQL, відстеження їх іноді може бути проблемою.

Огляд СКБД MongoDB

MongoDB – відкрита система управління базами даних (СУБД), яка використовує документно-орієнтовану модель бази даних, яка підтримує різні форми даних [19]. Це одна з численних нереляційних технологій бази даних, які виникли в середині 2000-х років під банером NoSQL для

використання у великих додатках даних та інших робочих місцях обробки, що містять дані, які не вписуються в жорстку реляційну модель. Замість використання таблиць і рядків, як у реляційних базах даних, архітектура MongoDB складається з колекцій і документів.

Запис в MongoDB є документом, який є структурою даних, що складається з пар поля і значення. Документи MongoDB подібні до об'єктів JavaScript Object Notation, але використовують варіант Binary JSON (BSON), який вміщує більше типів даних [20]. Поля в документах близькі до стовпців реляційної бази даних, а значення, які вони містять, можуть бути різними типами даних, включаючи інші документи, масиви і масиви документів, згідно з керівництвом користувача MongoDB.

Документи, які також повинні включати первинний ключ як унікальний ідентифікатор, є основною одиницею даних у MongoDB. Колекції містять набори документів і функціонують як еквівалент реляційних таблиць бази даних. Колекції можуть містити будь-який тип даних, але обмеження полягає в тому, що дані в колекції не можуть бути розподілені між різними базами даних.

Оболонка mongo – це інтерактивний інтерфейс JavaScript для MongoDB [21], який дозволяє користувачам запитувати і оновлювати дані, а також проводити адміністративні операції. Оболонка є стандартним компонентом розподілів з відкритим вихідним кодом MongoDB. Після встановлення MongoDB користувачі підключають оболонку mongo до запущених MongoDB-екземплярів.

Формат зберігання та обміну даними BSON, що використовується в MongoDB, забезпечує двійкове представлення документів, подібних до JSON. Автоматична функція sharding – це ще одна ключова особливість, яка дозволяє розподіляти дані в колекції MongoDB по декількох системах для горизонтальної масштабованості, оскільки збільшуються обсяги даних і вимоги до пропускної здатності.

СУБД NoSQL використовує єдину архітектуру майстра для

узгодженості даних, з вторинними базами даних, які підтримують копії первинної бази даних. Операції автоматично копіюються до цих вторинних баз даних для автоматичного переключення.

Переваги СКБД MongoDB:

- підтримує пошук за полями, регулярні пошуки виразів і пошукові запити;
- будь-яке поле в документі BSON можна індексувати;
- забезпечує високу доступність за допомогою наборів реплік, що складається з двох або більше копій вихідних даних;
- дозволяє sharding. Це метод, який дозволяє MongoDB масштабувати по горизонталі, тобто дані будуть розподілені і розділені на діапазони, а потім збережені в різних осколках, які можуть бути розташовані на різних серверах. Клавіші Shard використовуються для визначення розподілу даних;
- MapReduce може застосовуватися для включення пакетної обробки даних, а також для виконання операцій агрегації;
- MongoDB може використовуватися як файлова система, яка використовує вищезгадані функції і діє розподіленим способом через sharding.

Недоліки СКБД MongoDB:

- MongoDB не підтримує приєднання, як реляційна база даних. Проте можна скористатися функціональними можливостями об'єднання, додавши їх вручну. Але це може уповільнити виконання і вплинути на продуктивність;
- СКБД зберігає імена ключів для кожної пари значень. Крім того, через відсутність функціональних можливостей об'єднання, існує надмірність даних. Це призводить до збільшення непотрібного використання пам'яті;
- неможливо виконувати вкладання документів для більш ніж на 100 рівнів;

- розмір документа – не більше 16 МБ.

Порівняльний систем керування базами даних з точки зору вимог до системи

З точки зору вимог до системи, система керування базами даних має вирішувати такі задачі: стійкість до масштабування, безпека зберігання даних у базі, швидкодія, гнучкість.

Система керування базами даних MySQL не може до кінця забезпечити такі параметри як стійкість до масштабування та безпека зберігання даних у базі [22], адже має слабкіші характеристики, порівняно з конкурентами. Гнучкість СКБД є недостатньою для виконання поставлених задач, адже MySQL може бути застосована лише до програм з жорсткою реляційною моделлю бази даних [23].

Система керування базами даних MongoDB може забезпечити усі поставлені вимоги, адже є більш прогресивною, порівняно з конкурентами. Через те, що існує комьюніті розробників, які вдосконалюють СКБД кожного дня. Гнучкість та швидкодія забезпечуються за рахунок того, що база даних є документно-орієнтованою і не вимагає від програми жорсткої реляційної моделі.

2.3. Висновки

У цьому розділі були розглянуті та проаналізовані основні інструменти розробки десктопних додатків. Отримана інформація була систематизована з метою використання в подальшій роботі для визначення конкретних засобів реалізації, що будуть використовуватись при розробці автоматизованої системи обліку та контролю виробництва паливних брикетів RUF.

Для розробки програмного забезпечення було вирішено обрати мову програмування JavaScript, оскільки однією з основних задач майбутнього програмного забезпечення є кросплатформеність. JavaScript може забезпечити це за допомогою фреймворка, що має назву Electron. Він

дозволяє розбити надійні графічні додатки для настільних операційних систем з використанням веб-технологій. Фреймворк включає в себе Node.js [12], для роботи з back-end, і бібліотеку рендерінга з Chromium. Тобто розроблене програмне забезпечення задіє в собі всі найкращі веб-інструменти, при чому буде працювати на десктопних операційних системах і підтримуватиме кросплатформеність. Таку потужність не може запропонувати жодна з розглянутих вище мов. Саме тому вибір мови програмування JavaScript є оптимальним для вирішення поставлених у дипломній роботі задач.

В якості системи керування базами даних було обрано документно-орієнтовану СКБД MongoDB, оскільки вона використовується майже у всіх великих додатках, що містять дані, які не вписуються в жорстку реляційну модель. Також СКБД має чітку інструкцію для розробників, є простою у встановленні та використанні, має всі необхідні функції, та демонструє високу швидкість виконання команд.

3. РОЗРОБКА СИСТЕМИ

3.1. Загальний опис системи

Після огляду та аналізу аналогічних систем, було вирішено розробити систему обліку та контролю виробництва паливних брикетів RUF.

Розроблена система є настільним комп'ютерним додатком з інтуїтивно зрозумілим інтерфейсом та багатьма функціональними можливостями.

Діаграма використання представлена на рис. 3.

Першим, що бачить користувач при вході у систему є головне меню. У ньому містяться такі пункти:

- Контроль виробництва брикетів;
- Аналітичний блок;
- Контроль стану обладнання.

У вікні «*Контроль виробництва брикетів*» є 3 умовні блоки:

- Стан бункера з сировиною. У цьому блоці міститься інформація про кількість сировини у бункері, а також форма для внесення інформації про додавання сировини у бункер. Списання сировини з бункера буде відбуватися автоматично під час запуску лінії.
- Виробничі процеси. У цьому блоці міститься форма для введення даних про кількість сировини, з якої будуть робитися паливні брикети RUF. Також там буде знаходитися кнопка «Запустити лінію». Ця кнопка буде натискатися оператором одночасно з запуском лінії для візуалізації виробничих процесів. Візуалізація буде відбуватися через індикатори з показниками часу та проробленої роботи у відсотках.
- Стан складу з готовою продукцією. У цьому блоці буде міститися інформація про кількість готової продукції на складі. Інформація буде оновлюватися автоматично після завершення роботі ліній

виробництва брикетів. Списання готової продукції зі складу буде проходити через форму, у яку вводяться дані про кількість продукції, яка має бути списана.

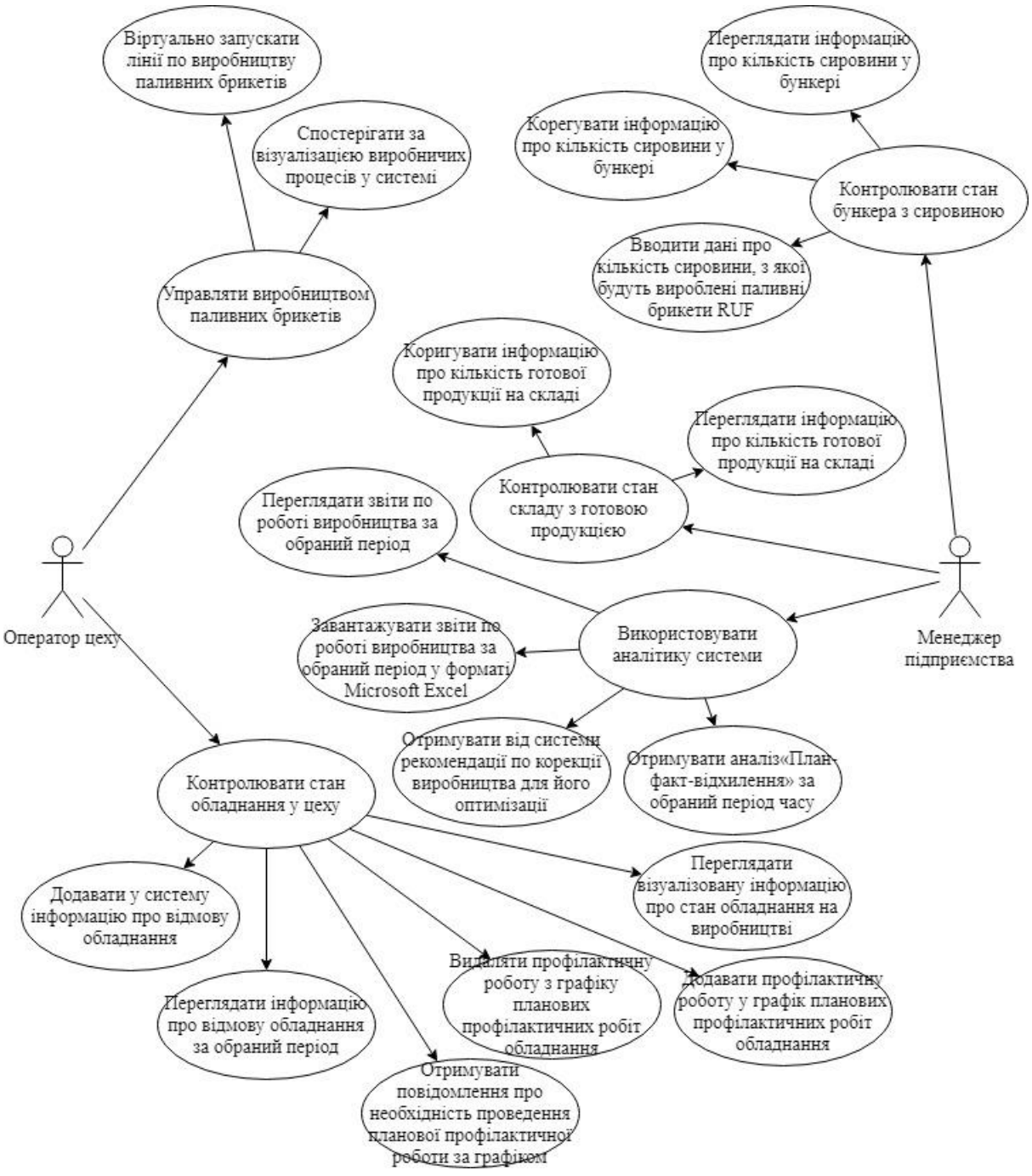


Рис. 3. Діаграма використання

У вікні «*Аналітичний блок*» є 3 умовні блоки:

- Звіти по роботі виробництва. У цьому блоці містяться звіти з усіма показниками виробництва. Звіти по обраному періоду можуть бути вигружені з системи у форматі Microsoft Excel для подальшого аналізу сторонніми спеціалістами.
- Аналіз «План-факт-відхилення». У цьому блоці можна обрати період для аналізу. Після чого на екран виведуться дані з системи про цей період. Користувач зможе ввести фактичні дані зі складу та побачити відхилення, якщо таке є.
- Рекомендовані коригувальні дії. На основі звітів з системи, користувач має змогу створити рекомендації по корекції виробництва для його оптимізації.

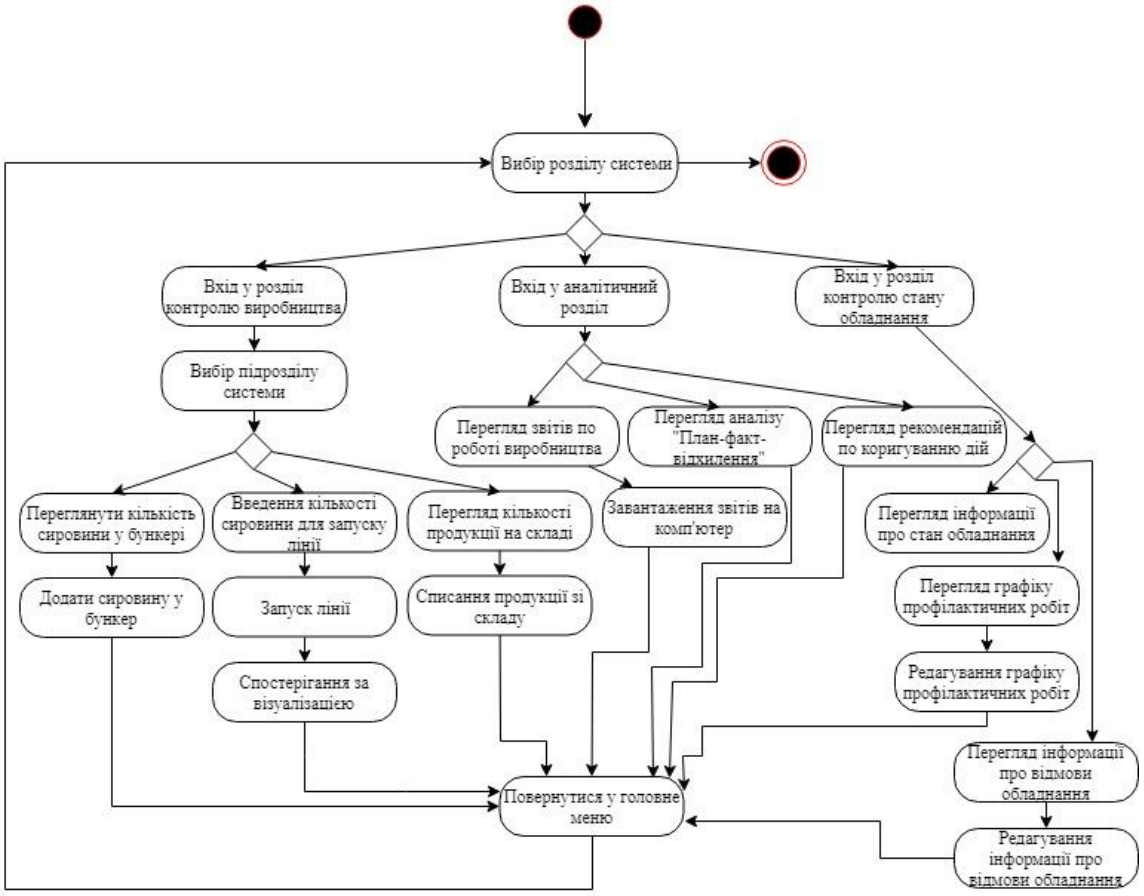


Рис. 4. Алгоритм використання програми

У вікні «*Контроль стану обладнання*» є 3 умовні блоки:

- Інформація про стан обладнання. У цьому блоці міститиметься інформація з наглядними індикаторами з показниками у відсотках та реальних числах про стан обладнання на виробництві.
- Графік планових профілактичних робіт. У цьому блоці користувач зможе додати та видалити роботу про профілактику обладнання. Коли час профілактики прийде, система видасть користувачу повідомлення з нагадуванням.
- Інформація про відмови обладнання. У цьому блоці можна є спеціальна форма для додавання інформації про відмову обладнання. Інформація про всі попередні відмови може бути виведена відповідно до обраного періоду.

Алгоритм використання програми представлений на рис. 4.

3.2. Функціональні вимоги до системи

Користувач системи повинен мати наступні можливості:

- переглядати інформацію про кількість сировини у бункері;
- корегувати інформацію про кількість сировини у бункері;
- вводити дані про кількість сировини, з якої будуть вироблені паливні брикети RUF;
- віртуально запускати лінії по виробництву паливних брикетів;
- спостерігати за візуалізацією виробничих процесів у системі;
- переглядати інформацію про кількість готової продукції на складі;
- коригувати інформацію про кількість готової продукції на складі;
- переглядати звіти по роботі виробництва за обраний період;
- завантажувати звіти по роботі виробництва за обраний період у форматі Microsoft Excel;
- отримувати аналіз «План-факт-відхилення» за обраний період часу;

- отримувати від системи рекомендації по корекції виробництва для його оптимізації;
- переглядати візуалізовану інформацію про стан обладнання на виробництві;
- додавати профілактичну роботу у графік планових профілактичних робіт обладнання;
- видаляти профілактичну роботу з графіку планових профілактичних робіт обладнання;
- отримувати повідомлення про необхідність проведення планової профілактичної роботи за графіком;
- додавати у систему інформацію про відмову обладнання;
- переглядати інформацію про відмову обладнання за обраний період.

3.3. Архітектура системи

Архітектура програми

Програмне забезпечення було створено на основі клієнт-серверної архітектурної моделі.

У Electron процес, який виконує основний скрипт package.json, є головним процесом. Сценарій, який виконується в основному процесі, відображає графічний інтерфейс, створюючи веб-сторінки. Розроблене програмне забезпечення має один основний процес.

Оскільки Electron використовує Chromium для відображення веб-сторінок [14], про розробці системи обліку та контролю паливних брикетів RUF була використана багатопроцесна архітектура Chromium. Кожна веб-сторінка в запускається у власному процесі, який називається процесом рендеринга.

Було використано API Node.js у веб-сторінках, що дозволило взаємодіяти з операційною системою нижчого рівня.

Основний процес створює веб-сторінки, використовуючи екземпляри

BrowserWindow. Кожен екземпляр BrowserWindow запускає веб-сторінку у власному процесі візуалізації. Коли екземпляр BrowserWindow руйнується, відповідний процес рендеринга також припиняється.

Основний процес керує всіма веб-сторінками та їх відповідними процесами візуалізації. Кожен процес візуалізації виділяється і управляє роботою веб-сторінки.

На веб-сторінках виклик рідних інтерфейсів API, пов'язаних з графічним інтерфейсом, не відбувається. Керування власними ресурсами графічного інтерфейсу на веб-сторінках є небезпечним і може спричинити виток ресурсів. Тому для виконання операції графічного інтерфейсу на веб-сторінці, процес візуалізації веб-сторінки обмінюється даними з основним процесом, де робить запит, щоб основний процес виконував ці операції.

Всі вікна були створені за допомогою класу BrowserWindow. Він доступний лише в основному процесі. Оскільки зв'язок між процесами можливий, процес візуалізації може викликати основний процес для виконання завдань. Електрон був підключений з модулем [15], який називається пультом дистанційного керування, який надає API, як правило, тільки в основному процесі.

Електрон надає повний доступ до Node.js як в основному, так і в процесі візуалізації. Усі API, доступні в Node.js, доступні в Електроні. Модулі Node.js були використані під час розробки системи. Було обрано модуль npm.

Схема бази даних

Для створення бази даних була обрана система контролю базами даних MongoDB. Це інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безшовне оточення для систем баз даних.

Можливості системи керування базами даних MongoDB:

- підтримка спеціальних запитів. У MongoDB, ви можете шукати по полю, робити суворий запит, а також пошук регулярних виразів;
- індексація. Можна індексувати будь-яке поле документа;
- реплікація. MongoDB підтримує реплікацію Master Slave. Майстер може виконувати читання та запис, а дані про підлеглих копіює з майстра і може використовуватися лише для читання або резервного копіювання;
- дублювання даних. MongoDB може працювати на декількох серверах. Дані дублюються для того, щоб зберегти систему, а також зберегти її стан під час збою обладнання;
- балансування навантаження. СКБД має автоматичну конфігурацію балансування навантаження через дані, розміщені в шардах;
- підтримує map reduce та агрегацію;
- використовує JavaScript замість процедур;
- забезпечує високу продуктивність;
- зберігає файли будь-якого розміру легко, не ускладнюючи стек;
- легко керувати у випадку збоїв;
- підтримується модель даних JSON з динамічними схемами;
- забезпечується автоматичне масштабування для горизонтальної масштабованості;
- має вбудовану реплікація для високої доступності.

Основні документи зі схеми бази даних розробленого програмного забезпечення:

- bunker – містить інформацію про бункер;
- process – містить інформацію про один запуск виробничих ліній. Зберігає інформацію про усі параметри, з якими лінія була запущена та відпрацювала;

- manufacturing – зберігає інформацію про усі запуски ліній та є головним документом;
- storage – містить інформацію про склад з готовою сировиною;
- machine – зберігає інформацію про одиницю обладнання у цеху;
- equipment – містить інформацію про усе обладнання виробничого цеху;
- prevention schedule – зберігає інформацію з графіком профілактичних робіт для обладнання у цеху;
- prevention – зберігає інформацію про профілактичну роботу у цеху;
- equipment bugs – містить інформацію про відмови обладнання у цеху.

Схема бази даних представлена на рис. 5.

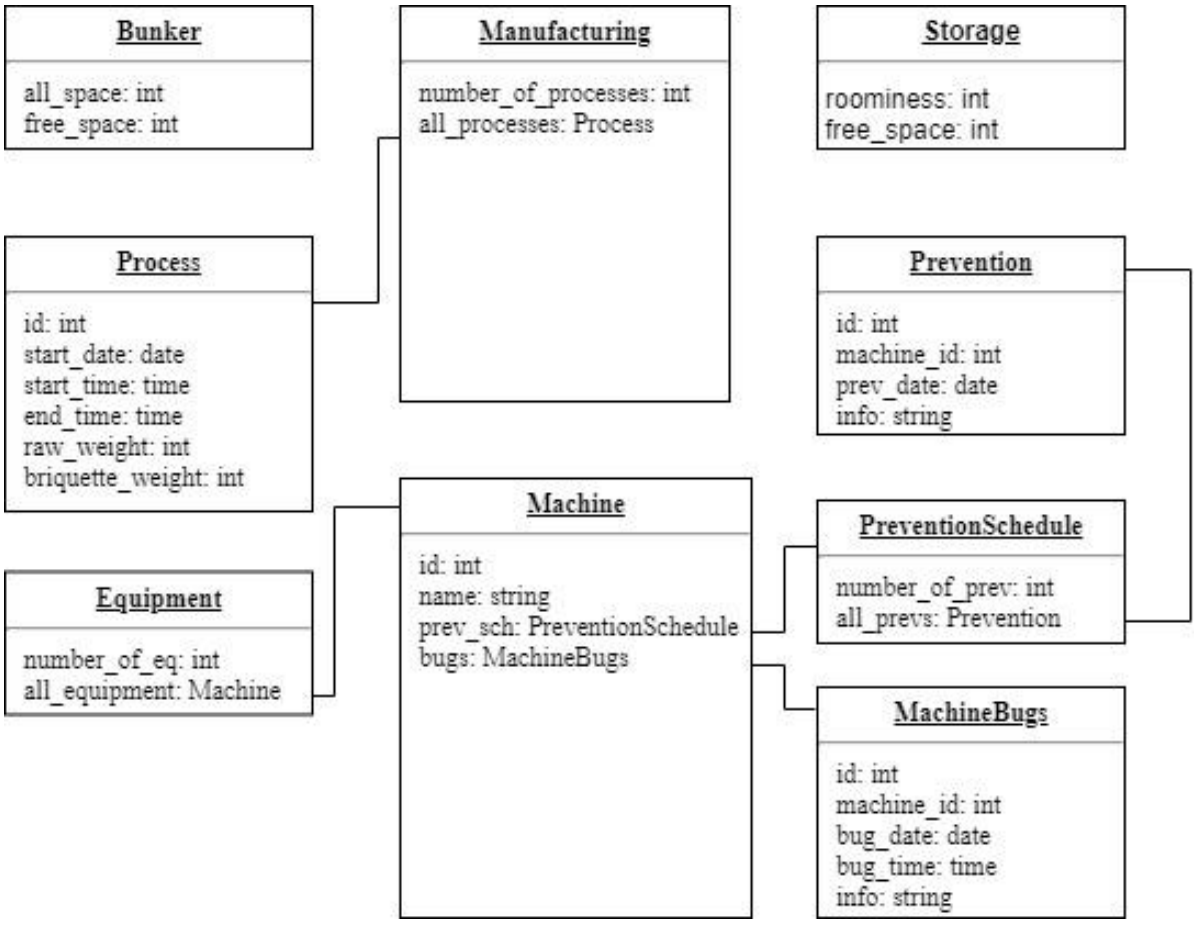


Рис. 5. Схема бази даних

3.4. Висновки

В даному розділі був проведений загальний опис розробленого програмного забезпечення, були перелічені висунуті функціональні вимоги до системи. Дані вимоги в повній мірі були реалізовані у системі.

Також були розглянуті використані технології, зокрема використання фреймворку Electron, що є одним з найпопулярніших, найбільш використовуваних та ефективних інструментів для розроблення сучасних кросплатформених настільних програмних застосунків.

Для реалізації серверної частини програми був використаний фреймворк NodeJS, що надає можливість JavaScript взаємодіяти з пристроями введення-виведення через свій API (написаний на C++), підключати інші зовнішні бібліотеки, написані на різних мовах, забезпечуючи виклики до них з JavaScript-коду.

Для вигляду вікон додатку був використаний сучасний набір технологій, таких як мова розмітки HTML і використання стилів CSS.

Також у розділі було детально розглянуто документно-орієнтовану систему керування базами даних MongoDB. Було реалізовано і представлено головні документи бази даних з детальним їх описом.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Аналіз реалізації вимог

При створенні програмного забезпечення було поставлено і виконано наступні вимоги:

- функціональність (відповідність задачам користувача). Вимога реалізована за рахунок розробки функціональних блоків до кожного запиту користувача;
- зрозумілий користувачу інтерфейс. Реалізовано через використання мови розмітки HTML, CSS та основ UI та UX;
- адаптивність до вирішення поставленої задачі. Реалізація відбувається через зручне меню, що дозволяє користувачу перемикатися між задачами;
- простота експлуатації, максимальний час навчання персоналу – 3 робочі дні. Вимога забезпечена за рахунок реалізації максимально простого інтерфейсу, який є зрозумілим користувачу на інтуїтивному рівні;
- використання з ноутбуком або ПК. Реалізовано за допомогою використання двигуна Chromium, що дозволяє запускати JavaScript код на комп'ютері;
- операційна система Windows, Mac OS або Linux. Вимога реалізована за допомогою роботи з фреймворком Electron, що дозволяє розробляти настільні кросплатформені додатки;
- стійкість системи до масштабування. Забезпечено завдяки використанню документно-орієнтованої бази даних, що дозволяє шардинг;
- безпека зберігання даних у базі. Реалізовано завдяки використанню найбільш прогресивної та гнучкої СКБД MongoDB;

- стійкість до хакерських атак. Забезпечено завдяки внутрішньому захисту бази даних від сторонніх ін’єкцій;
- швидкодія системи, максимальний час відгуку – 1 секунда. Реалізовано шляхом використання асинхронної мови програмування JavaScript, що дозволяє виконувати декілька процесів одночасно;
- збереження у базу всіх даних, введених в систему. Вимога реалізована завдяки розробленню оптимальної архітектури системи та стійкому зв’язку з базою даних;
- максимальний час оновлення екрана – 3 секунди. Забезпечено шляхом використання клієнт-серверної архітектури програми. Оновлення екрану відбувається на клієнтській частині програмного застосунку, майже не задіюючи сервер;
- самостійне відновлення роботи системи після помилки. Вимога реалізована за рахунок використання фреймворка Node.js, що забезпечує стійкість сервера.

Всі вимоги [25] до програмного забезпечення реалізовані, розробка може вважатися успішною [26].

4.2. Тестування системи

Тестування програми є невід’ємною частиною процесу створення закінченого програмного продукту [27] та є однією з основних стадій життєвого циклу програмного забезпечення.

Тестування програмного забезпечення можна визначити, оскільки процес верифікації та валідації того, що програмне забезпечення чи додаток не містить помилок, відповідає технічним вимогам, керованим розробкою та відповідає вимогам користувача, ефективно обробляючи всі виняткові та граничні випадки [28].

Процес тестування програмного забезпечення спрямований не тільки на виявлення помилок у існуючому програмному забезпеченні, але й на

пошук заходів для вдосконалення програмного забезпечення з точки зору ефективності, точності та зручності використання. В основному він призначений для вимірювання специфікації, функціональності та продуктивності програмного забезпечення або додатку.

Під час тестування системи обліку та контролю виробництва паливних брикетів RUF використовувалось 2 підходи.

Першим способом тестування є ручне тестування, за допомогою якого вводилися різні дані до текстових полів, зокрема некоректні дані, дані великого об’єму або зовсім не вводилися дані. За результатами цього тестування у проект були додані повідомлення про відповідні помилки, які виникають при вводі некоректних даних користувачами. За допомогою тестування була перевірена коректність підключення до СКБД MongoDB та відслідковувалися дані, що заносилися до СКБД.

Другим видом тестування було автоматизоване юніт-тестування. Для такого типу тестування використовувався спеціалізований фреймворк Mocha [29]. Mocha – це багатофункціональний фреймворк для тестування програм, написаних на мові JavaScript. Він працює на Node.js і в браузері, що робить асинхронне тестування значно простішим. Тести Mocha виконуються послідовно, дозволяючи гнучку і точну звітність, одночасно зіставляючи невикористані винятки з правильними тестовими випадками. Кожна функція виконує певне завдання. Щоб перевірити функцію, її потрібно викликати з тестового файлу або файлу специфікації з необхідними входами. Потім створюється твердження для перевірки результату або завдання функції. Отриманий результат автоматично порівнюється зі створеним розробником заздалегідь [30]. Якщо результати збігаються, то функція вважається протестованою та працюючою правильно.

За допомогою тестування були отримані важливі результати щодо вдосконалення та доопрацювання програми. Так як тестування проводилось неперервним циклом на кожному етапі розробки, усі

результати доопрацювань були враховані. Це дало змогу розробити надійне програмне забезпечення, що задовольняє усім поставленим вимогам.

4.3. Порівняння розробки з існуючими аналогами

В першому розділі дипломного проекту було розглянуто різноманітні аналогів, що стосуються управління виробництвом.

Всі аналоги були вивчені з метою виявлення основних переваг та недоліків вже існуючих рішень.

Розроблене програмне забезпечення відповідає усім визначеним вимогам на основі досліджених переваг аналогів, а саме:

- зручний інтерфейс;
- простота використання;
- надійність;
- стійкість до хакерських атак.

Окрім реалізації виявлених переваг, при розробці веб-застосунку були усунуті основні недоліки аналогічних систем, такі як:

- масштабність. Розглянуті аналоги мають велику кількість функцій, які не будуть використані для управління виробництвом паливних брикетів RUF;
- складність налаштування. Налаштування великої ERP-системи, якими і є розглянуті аналоги, потягне за собою великі витрати ресурсів, таких як грошовий, людський та часовий.
- вартість супроводу. Для супроводу роботи великої ERP-системи підприємству доведеться або щомісячно сплачувати велику суму службі підтримки (у випадку з Бітрікс24) або наймати висококваліфікованих фахівців (у випадку С1), які потребують заробітної плати відповідного рівня.
- широка спеціалізація. Розглянуті аналоги не мають чіткої спеціалізації на виробництві паливних брикетів RUF.

Проаналізувавши розроблене програмне забезпечення, можна стверджувати, що воно є достойним конкурентом вже існуючим на ринку аналогам. Адже автоматизована система обліку та контролю паливних брикетів RUF має чітку спеціалізацію і заточена під конкретне виробництво. Цей варіант є найоптимальнішим, адже не вимагає витрати надлишкових ресурсів і в той самий час повністю забезпечує усі потреби підприємства.

4.4. Рекомендації щодо подальшого вдосконалення

Розроблене програмне забезпечення може використовуватися у реальному житті, адже задовольняє усім вимогам і може взяти на себе управління виробництвом, замінивши ручне управління. Проте, у сучасному світі з стрімко зростаючими запитами користувачів, підприємство має бути гнучким і швидко підлаштовуватися під ситуацію.

Отже, у майбутньому програмне забезпечення можна вдосконалювати шляхом додання нових можливостей, що будуть спрощувати управління підприємством за допомогою автоматизованої системи обліку та контролю виробництва паливних брикетів RUF:

- розділяння брикетів за видом деревини (дуб, сосна, бук і т.д.);
- сортування брикетів по параметру вологості;
- сортування брикетів по параметру форми;
- ускладнення аналітичного блоку;
- створення більш повних звітів зі складною структурою;
- підключення декількох цехів одночасно;
- можливість робити вибірки у звітах не тільки по параметру дати запуску лінії, а і по іншим параметрам (тривалість роботи, кількість сировини, кількість вихідної продукції);
- можливість робити вибірки у аналізі «План-факт-відхилення» не тільки по параметру періоду роботи виробництва, а і по іншим параметрам (сорт деревини, вологість брикетів, форма брикетів);

- створення мобільної версії додатку;
- створення браузерної версії додатку;
- створення резервних копій даних у хмарному сховищі;
- покращення властивостей інтерфейсу;
- підключення модулю для ведення бухгалтерії підприємства;
- підключення CRM-системи для оптимізації управління клієнтською базою та взаємодії з нею.

Усі ці покращення можуть бути легко впроваджені з часом, адже система спроектована таким чином та на такому рівні інкапсуляції, що підключення нових модулів майже не вплине на швидкодію та роботу вже реалізованих частин системи.

4.5. Висновки

В даному розділі було проведено аналіз реалізації вимог до розробленої програмної системи, описано процес тестування програмного забезпечення, виконано детальне порівняння розробки з існуючими аналогами, а також сформовано рекомендації щодо подальшого вдосконалення.

Під час аналізу реалізації вимог було детально описано кожен пункт висунутих вимог, а також спосіб та процес їх програмної реалізації. Після проведення аналізу вдалося зробити висновок, що усі поставлені вимоги були реалізовані у повній мірі.

Тестування програмного забезпечення проводилося двома способами: ручним (мануальним) та автоматизованим, за допомогою методу юніт-тестування. За результатами мануального тестування були виявлені помилки, які виникають при вводі некоректних даних користувачами, після чого у проект були додані повідомлення про відповідні помилки. Для автоматизованого способу тестування використовувався спеціалізований фреймворк Mocha. За допомогою юніт-тестування були отримані важливі результати щодо вдосконалення та

доопрацювання програми. Так як тестування проводилось неперервним циклом на кожному етапі розробки, усі результати доопрацювань були враховані.

У ході порівняння розробленої програмної системи з існуючими аналогами було виявлено, що вона є достойним конкурентом вже існуючим на ринку аналогам, адже має чітку спеціалізацію і створена для конкретного виробництва. Цей варіант є найоптимальнішим, адже не вимагає витрати надлишкових ресурсів і в той самий час повністю забезпечує усі потреби підприємства.

Сформовані рекомендації щодо подальшого вдосконалення можуть бути легко впроваджені з часом, адже система спроектована таким чином та на такому рівні інкапсуляції, що підключення нових модулів майже не вплине на швидкодію та роботу вже реалізованих.

ВИСНОВКИ

Метою даного дипломного проекту було розроблення програмної системи, яка виконувала б облік та контроль виробництва паливних брикетів RUF. Система була розроблена для конкретного підприємства, але є масштабованою і може бути застосована на інших підприємствах такого типу.

Після проведення аналізу існуючих аналогів, було зроблено висновок, що система має бути розроблена у вигляді кросплатформеного настільного додатку. Він повинен працювати на операційних системах Windows, Linux і MacOS, бути стійким, потужним і мати зручний та зрозумілий інтерфейс для робітників підприємства.

На основі проведеного аналізу щодо вибору засобів реалізації, було вирішено створювати настільний додаток на мові Java Script з допомогою фреймворків Node.js та Electron, використовуючи систему керування базами даних MongoDB.

Розроблений веб-додаток надає користувачу змогу:

- управляти інформацією про сировину, а саме переглядати інформацію про кількість сировини у бункері, корегувати інформацію про кількість сировини у бункері, вводити дані про кількість сировини, з якої будуть вироблені паливні брикети RUF;
- спостерігати за візуалізованими виробничими процесами та віртуально запускати лінії по виробництву паливних брикетів;
- корегувати інформацію про кількість готової продукції на складі;
- мати швидкий доступ до аналітики виробництва, тобто переглядати звіти по роботі виробництва за обраний період, завантажувати звіти по роботі виробництва за обраний період у форматі Microsoft Excel, отримувати аналіз «План-факт-відхилення» за обраний період часу, отримувати від системи рекомендації по корекції виробництва для його оптимізації;

- у режимі реального часу стежити за станом виробничого обладнання, а саме переглядати візуалізовану інформацію про стан обладнання на виробництві, додавати профілактичну роботу у графік планових профілактичних робіт обладнання, видаляти профілактичну роботу з графіку планових профілактичних робіт обладнання, отримувати повідомлення про необхідність проведення планової профілактичної роботи за графіком, додавати у систему інформацію про відмову обладнання, переглядати інформацію про відмову обладнання за обраний період.

Розробка програми виконана у повному обсязі та відповідає поставленим вимогам. Тестування настільного додатку виконано у відповідності до затвердженої програми та методики тестування.

Використання розробленого настільного додатку дозволить підприємству відійти від ручного управління виробництвом паливних брикетів RUF, що значно покращить роботу підприємства та дозволить вийти на новий рівень, порівняно з конкурентами.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Характеристика та особливості паливних брикет як палива [Електронний ресурс] – Режим доступу: http://www.biowatt.com.ua/informatsiya/harakteristika-ta-osoblivosti-palivnih-briket-yak-paliva/.
2. Виробництво паливних брикетів [Електронний ресурс]. – 2014. – Режим доступу: https://bio.ukr.bio/ua/articles/6015/.
3. Виготовлення паливних брикетів та гранул [Електронний ресурс]. – 2017. – Режим доступу: https://startup.ua/ua/projects/v-shark-15895.html.
4. Види обладнання для виробництва паливних брикетів [Електронний ресурс]. – 2017. – Режим доступу: http://greenpower.com.ua/clients/articles/2017-04-12-16-59-43//.
5. Обладнання для виробництва пелет та брикетів [Електронний ресурс]. – 2015. – Режим доступу: https://bio.ukr.bio/ua/articles/7541/.
6. JavaScript [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Learn/Getting_started_with_the_web/JavaScript_basics.
7. About JavaScript [Електронний ресурс]. – 2019. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.
8. What is Java technology and why do I need it? [Електронний ресурс] – Режим доступу: https://www.java.com/en/download/faq/whatis_java.xml.
9. C# (C Sharp) [Електронний ресурс] – Режим доступу: https://www.techopedia.com/definition/26272/c-sharp.
10. Node.js Introduction [Електронний ресурс] – Режим доступу: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm.

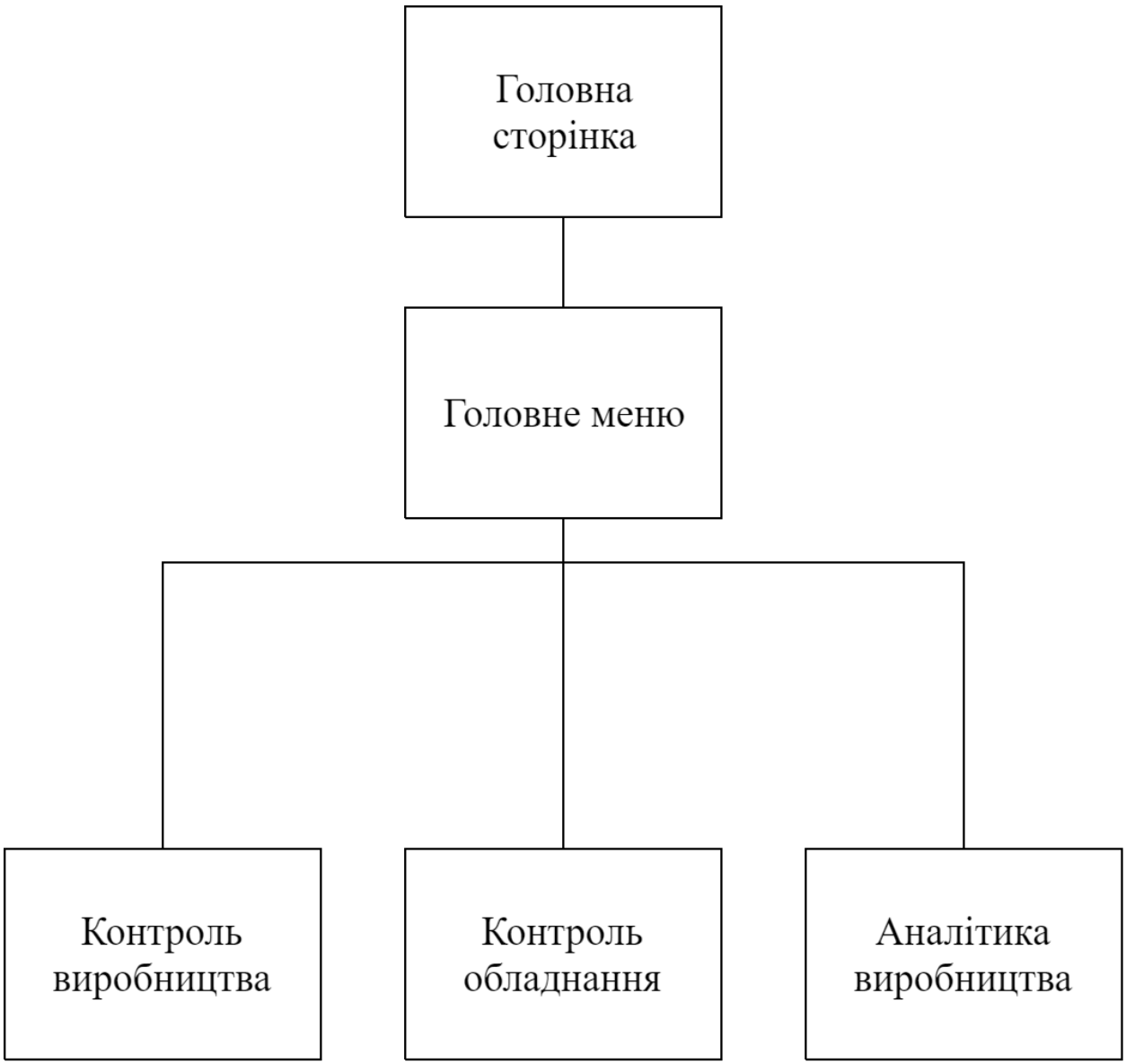
11. Про Node.js [Електронний ресурс] – Режим доступу: https://nodejs.org/uk/about/.
12. All About Node.Js You Wanted To Know [Електронний ресурс] – Режим доступу: https://codeburst.io/all-about-node-js-you-wanted-to-know-25f3374e0be7.
13. Electron Documentation [Електронний ресурс] – Режим доступу: https://electronjs.org/docs/tutorial/about.
14. Electron Apps [Електронний ресурс] – Режим доступу: https://electronjs.org/apps.
15. 4 must-know tips for building cross platform Electron apps [Електронний ресурс] – Режим доступу: https://blog.avocode.com/4-must-know-tips-for-building-cross-platform-electron-apps-f3ae9c2bffff.
16. Багатоплатформність [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Багатоплатформність.
17. Система керування базами даних [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Система_керування_базами_даних.
18. Класифікація СУБД [Електронний ресурс] – Режим доступу: https://stud.com.ua/50165/informatika/klasifikatsiya_subd.
19. MongoDB [Електронний ресурс] – Режим доступу: https://searchdatamanagement.techtarget.com/definition/MongoDB.
20. MongoDB co-creator explains why ‘NoSQL’ came to be, and why open source mastery is an elusive goal [Електронний ресурс] – Режим доступу: https://medium.com/s-c-a-l-e/mongodb-co-creator-explains-why-nosql-came-to-be-and-why-open-source-mastery-is-an-elusive-goal-3a138480b9cd.
21. The architecture of MongoDB NoSQL Database [Електронний ресурс] – Режим доступу: https://intellipaat.com/blog/what-is-mongodb/.

22. MySQL Documentation [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/.
23. How MySQL works [Електронний ресурс] – Режим доступу: https://searchoracle.techtarget.com/definition/MySQL.
24. Визначення та розробка вимог до ПЗ [Електронний ресурс] – Режим доступу: https://studopedia.com.ua/1_56231_viznachennya-ta-rozrobka-vimog-do-pz.html.
25. Особливості визначення вимог до програмного забезпечення [Електронний ресурс] – Режим доступу: https://cyberleninka.ru/article/n/osoblivosti-viznachennya-vimog-do-programnogo-zabezpechennya-ta-problemi-yih-analizu.
26. Аналіз вимог до програмного забезпечення [Електронний ресурс] – Режим доступу: https://kpi.stu.cn.ua/analiz-vimog-do-programnogo-zabezpech/.
27. Введення в тестування програмного забезпечення [Електронний ресурс] – Режим доступу: http://qalearning.com.ua/theory/lectures/material/testing-intro/.
28. Тестування програмного продукту [Електронний ресурс] – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm.
29. JavaScript—Unit Testing using Mocha and Chai [Електронний ресурс] – Режим доступу: https://codeburst.io/javascript-unit-testing-using-mocha-and-chai-1d97d9f18e71.
30. Unit Testing in JavaScript with Mocha [Електронний ресурс] – Режим доступу: https://www.taniarascia.com/unit-testing-in-javascript/.

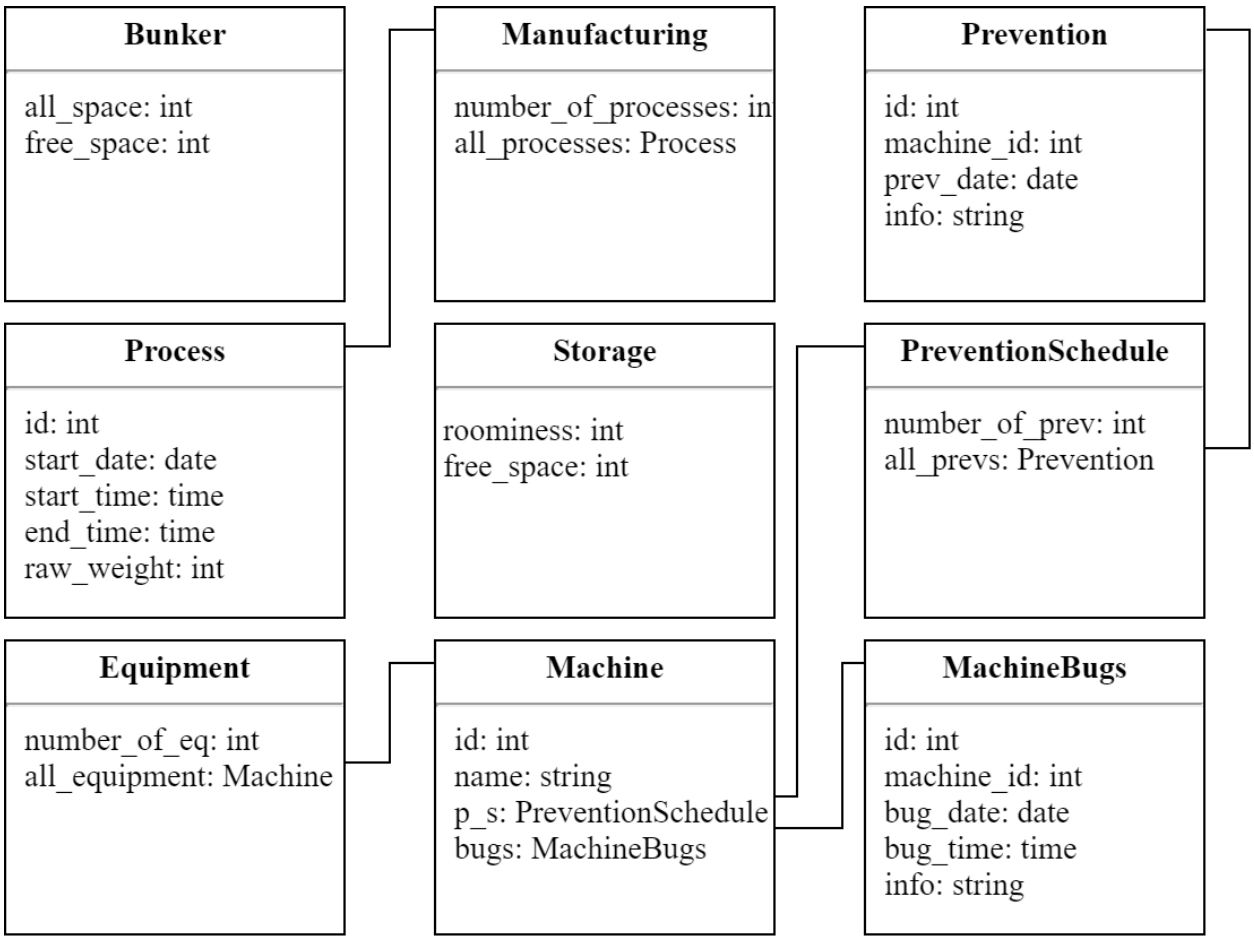
ДОДАТКИ

Додаток 1

Копії графічних матеріалів



ДП.045440-06-99
Програмна система для обліку та контролю виробництва паливних брикетів RUF.
Структура взаємодії представлень.
Схема структурна

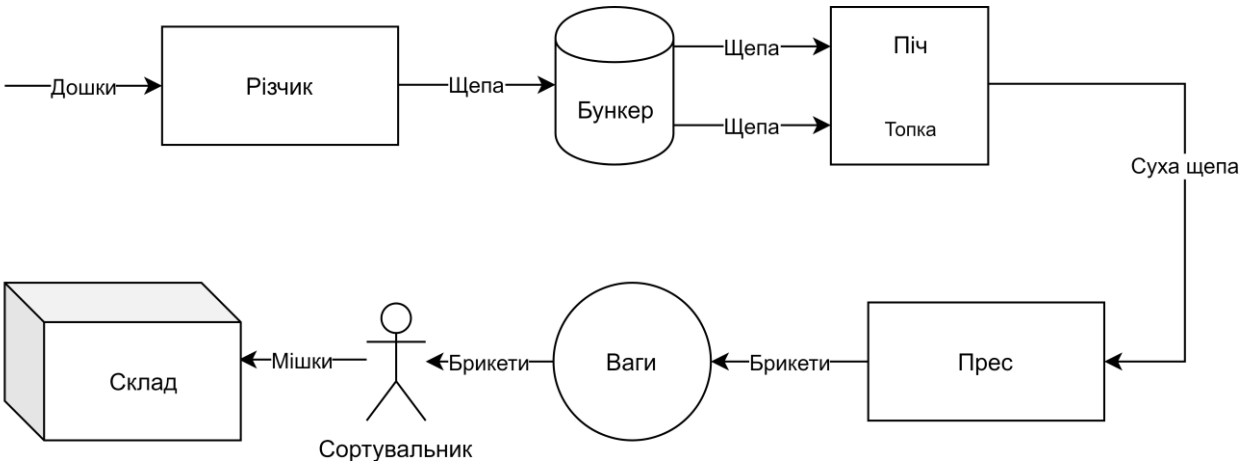


ДП.045440-07-99

Програмна система для обліку та контролю виробництва паливних брикетів RUF.

База даних системи. Схема даних


Ручна система управління виробництвом
Відсутні дані реального виробництва
Ручний контроль стану обладнання
Відсутні дані про виробничі запаси сировини
Відсутня база даних показників виробничих процесів
Відсутня інформація про кількість готової продукції
Відсутність інформації про стан обладнання
Відсутність графіку планових профілактичних робіт обладнання
Відсутність інформації про відмови обладнання
Відсутня аналітика виробництва
Неповні звіти по виробництву
Відсутність аналізу "План-факт-відхилення"
Відсутність можливості надати рекомендовані коригувальні дії


Дошки
Різчик
Щепа
Бункер
Щепа
Щепа
Піч
Топка
Суха щепа
Прес
Брикети
Ваги
Брикети
Сортувальник
Мішки
Склад

Додаток 2

Лістинги програм

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { materialAdded } from '../../../actions/bunker';

type Props = {};

class Bunker extends Component<Props> {
  props: Props;

  state = {
    inputValue: '',
    inputError: null
  };

  onInputChange = ev => this.setState({ inputValue: ev.target.value });

  onAddClick = () => {
    if (!this.state.inputValue.match(/^\d+$/)) {
      this.setState({ inputError: 'Значення має бути числом' });
      return;
    }
    const value = Number(this.state.inputValue);
    if (this.props.currentMaterial + value > this.props.capacity) {
      this.setState({ inputError: 'Введене значення переповнює бункер' });
      return;
    }
    this.setState({ inputError: null, inputValue: '' });
    this.props.materialAdded(value);
  };

  render() {
    return (
      <div>
        <div className="ui divider" />
        <h3
          style={{ textAlign: 'center', color: 'white' }}
```

```
          className="ui header"
        >
          Бункер з сировиною
        </h3>
        <div
          style={{
            display: 'flex',
            alignItems: 'center',
            justifyContent: 'space-between',
            margin: '0 100px'
          }}
        >
          <img
            style={{ width: 180 }}
            className="ui image"
src="http://i.piccy.info/i9/1b88b950e2e2f5d7e9b5dcc126001f14/1559906760/6897/1321379/17751.png"
          />
          <div style={{ width: 350 }}>
            <p>Сировини у бункері: {this.props.currentMaterial}т</p>
            <p>Вільно: {this.props.capacity - this.props.currentMaterial}т</p>
            <div className="ui form">
              <div className="field">
                <label style={{ color: 'grey' }}>Додати сировину</label>
                <input
                  type="text"
                  value={this.state.inputValue}
                  onChange={this.onInputChange}
                  name="first-name"
                  placeholder="Введіть кількість сировини"
                />
                {this.state.inputError && (
                  <p style={{ color: 'red' }}>{this.state.inputError}</p>
                )}
              </div>
              <button onClick={this.onAddClick} className="ui button">
                Додати
```

```
              </button>
            </div>
          </div>
        </div>

        <div className="ui divider" />
      </div>
    );
  }
}

const mapStateToProps = store => ({
  capacity: store.bunker.capacity,
  currentMaterial: store.bunker.currentMaterial
});

const mapDispatchToProps = {
  materialAdded
};

export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Bunker);

import * as React from 'react';

type Props = {
  children: React.Node
};

export default class App extends React.Component<Props> {
  props: Props;

  render() {
    const { children } = this.props;
    return <React.Fragment>{children}</React.Fragment>;
  }
```

```
}

import React, { Component } from 'react';
import { Provider } from 'react-redux';
import { ConnectedRouter } from 'connected-react-router';
import type { Store } from '../reducers/types';
import Routes from '../Routes';

type Props = {
  store: Store,
  history: {}
};

export default class Root extends Component<Props> {
  render() {
    const { store, history } = this.props;
    return (
      <Provider store={store}>
        <ConnectedRouter history={history}>
          <Routes />
        </ConnectedRouter>
      </Provider>
    );
  }
}

import { PRODUCT_DELETED } from '../actions/storage';
import type { Action } from './types';

const initialState = {
  capacity: 800,
  currentProduct: 150
};

export default function storage(state = initialState, action) {
  switch (action.type) {
    case PRODUCT_DELETED: {
      const { value } = action.payload;
```

```
      return {
        ...state,
        currentProduct: state.currentProduct - value
      };
    }
    default:
      return state;
  }
}

const initialState = {
  capacity: 1000,
  currentMaterial: 0
};

export default function bunker(state = initialState, action) {
  switch (action.type) {
    case MATERIAL_ADDED: {
      const { value } = action.payload;
      return {
        ...state,
        currentMaterial: state.currentMaterial + value
      };
    }
    default:
      return state;
  }
}

const history = createHashHistory();

const rootReducer = createRootReducer(history);

const configureStore = (initialState?: counterStateType) => {
  // Redux Configuration
  const middleware = [];
  const enhancers = [];
```

```
  // Thunk Middleware
  middleware.push(thunk);

  // Logging Middleware
  const logger = createLogger({
    level: 'info',
    collapsed: true
  });

  // Skip redux logs in console during the tests
  if (process.env.NODE_ENV !== 'test') {
    middleware.push(logger);
  }

  // Router Middleware
  const router = routerMiddleware(history);
  middleware.push(router);

  // Redux DevTools Configuration
  const actionCreators = {
    //...counterActions,
    ...routerActions
  };
  // If Redux DevTools Extension is installed use it, otherwise use Redux
compose
  /* eslint-disable no-underscore-dangle */
  const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__
    ? window.__REDUX_DEVTOOLS_EXTENSION_COMPOSE__({
        // Options: http://extension.remotedev.io/docs/API/Arguments.html
        actionCreators
      })
    : compose;
  /* eslint-enable no-underscore-dangle */

  // Apply Middleware & Compose Enhancers
  enhancers.push(applyMiddleware(...middleware));
  const enhancer = composeEnhancers(...enhancers);
```

```
  // Create Store
  const store = createStore(rootReducer, initialState, enhancer);

  if (module.hot) {
    module.hot.accept(
      '../reducers',
      // eslint-disable-next-line global-require
      () => store.replaceReducer(require('../reducers').default)
    );
  }

  return store;
};

export default class AppUpdater {
  constructor() {
    log.transports.file.level = 'info';
    autoUpdater.logger = log;
    autoUpdater.checkForUpdatesAndNotify();
  }
}

let mainWindow = null;

if (process.env.NODE_ENV === 'production') {
  const sourceMapSupport = require('source-map-support');
  sourceMapSupport.install();
}

if (
  process.env.NODE_ENV === 'development' ||
  process.env.DEBUG_PROD === 'true'
) {
  require('electron-debug')();
}

const installExtensions = async () => {
  const installer = require('electron-devtools-installer');
```

```
  const forceDownload = !!process.env.UPGRADE_EXTENSIONS;
  const extensions = ['REACT_DEVELOPER_TOOLS', 'REDUX_DEVTOOLS'];

  return Promise.all(
    extensions.map(name => installer.default(installer[name], forceDownload))
  ).catch(console.log);
};

/**
 * Add event listeners...
 */

app.on('window-all-closed', () => {
  // Respect the OSX convention of having the application in memory even
  // after all windows have been closed
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('ready', async () => {
  if (
    process.env.NODE_ENV === 'development' ||
    process.env.DEBUG_PROD === 'true'
  ) {
    await installExtensions();
  }

  mainWindow = new BrowserWindow({
    show: false,
    width: 1024,
    height: 728
  });

  mainWindow.loadURL(`file://${__dirname}/app.html`);

  // @TODO: Use 'ready-to-show' event
```

```
  //
https://github.com/electron/electron/blob/master/docs/api/browser-
window.md#using-ready-to-show-event
  mainWindow.webContents.on('did-finish-load', () => {
    if (!mainWindow) {
      throw new Error('"mainWindow" is not defined');
    }
    if (process.env.START_MINIMIZED) {
      mainWindow.minimize();
    } else {
      mainWindow.show();
      mainWindow.focus();
    }
  });

  mainWindow.on('closed', () => {
    mainWindow = null;
  });

  const menuBuilder = new MenuBuilder(mainWindow);
  menuBuilder.buildMenu();

  // Remove this if your app does not use auto updates
  // eslint-disable-next-line
  new AppUpdater();
});

export default class MenuBuilder {
  mainWindow: BrowserWindow;

  constructor(mainWindow: BrowserWindow) {
    this.mainWindow = mainWindow;
  }

  buildMenu() {
    if (
      process.env.NODE_ENV === 'development' ||
      process.env.DEBUG_PROD === 'true'
```

```
  ) {
    this.setupDevelopmentEnvironment();
  }

  const template =
    process.platform === 'darwin'
      ? this.buildDarwinTemplate()
      : this.buildDefaultTemplate();

  const menu = Menu.buildFromTemplate(template);
  Menu.setApplicationMenu(menu);

  return menu;
}

setupDevelopmentEnvironment() {
  this.mainWindow.openDevTools();
  this.mainWindow.webContents.on('context-menu', (e, props) => {
    const { x, y } = props;

    Menu.buildFromTemplate([
      {
        label: 'Inspect element',
        click: () => {
          this.mainWindow.inspectElement(x, y);
        }
      }
    ]).popup(this.mainWindow);
  });
}
```

Додаток З

Копія презентації

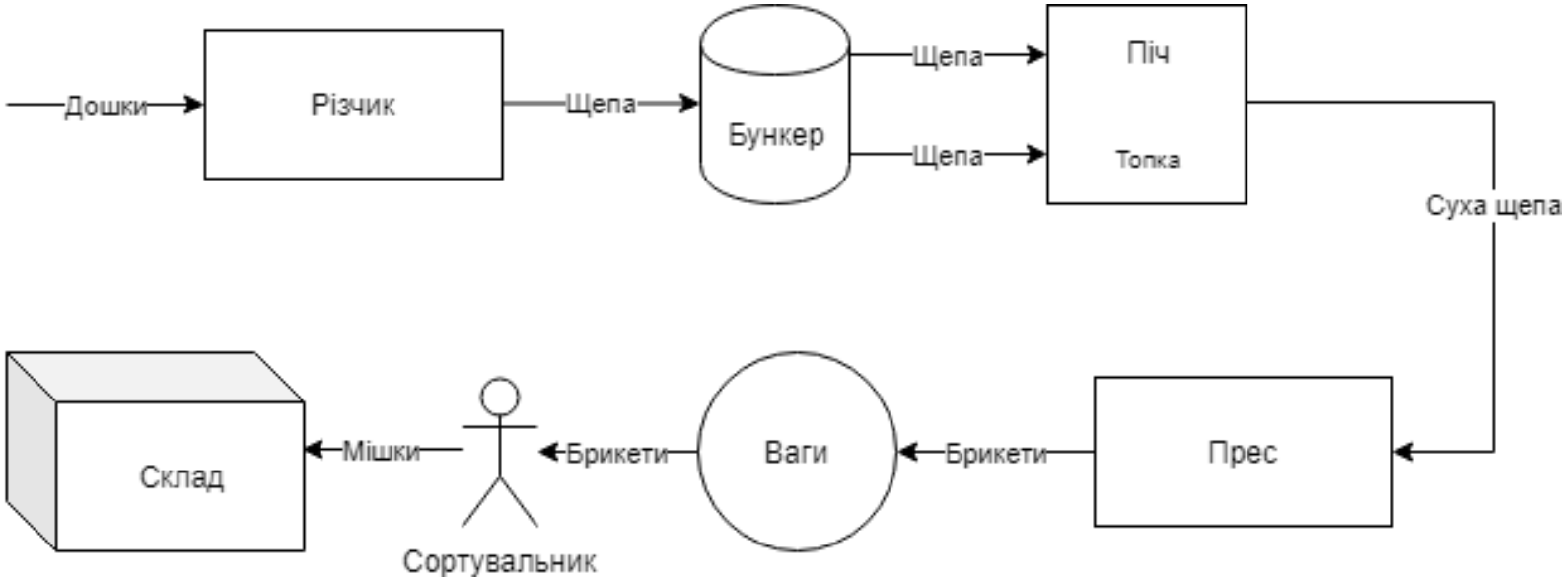
Програмна система для обліку та контролю виробництва паливних брикетів RUF

Виконала:
студентка групи КП-52 Пахомова Є.О.
Науковий керівник:
старший викладач, к.т.н., Люшенко Л.А.

Постановка задачі

Розробити програмну системи, що здійснює облік та контроль виробництва паливних брикетів RUF.

Схема виробничої лінії



Дерево проблем підприємства

Існуючі аналоги

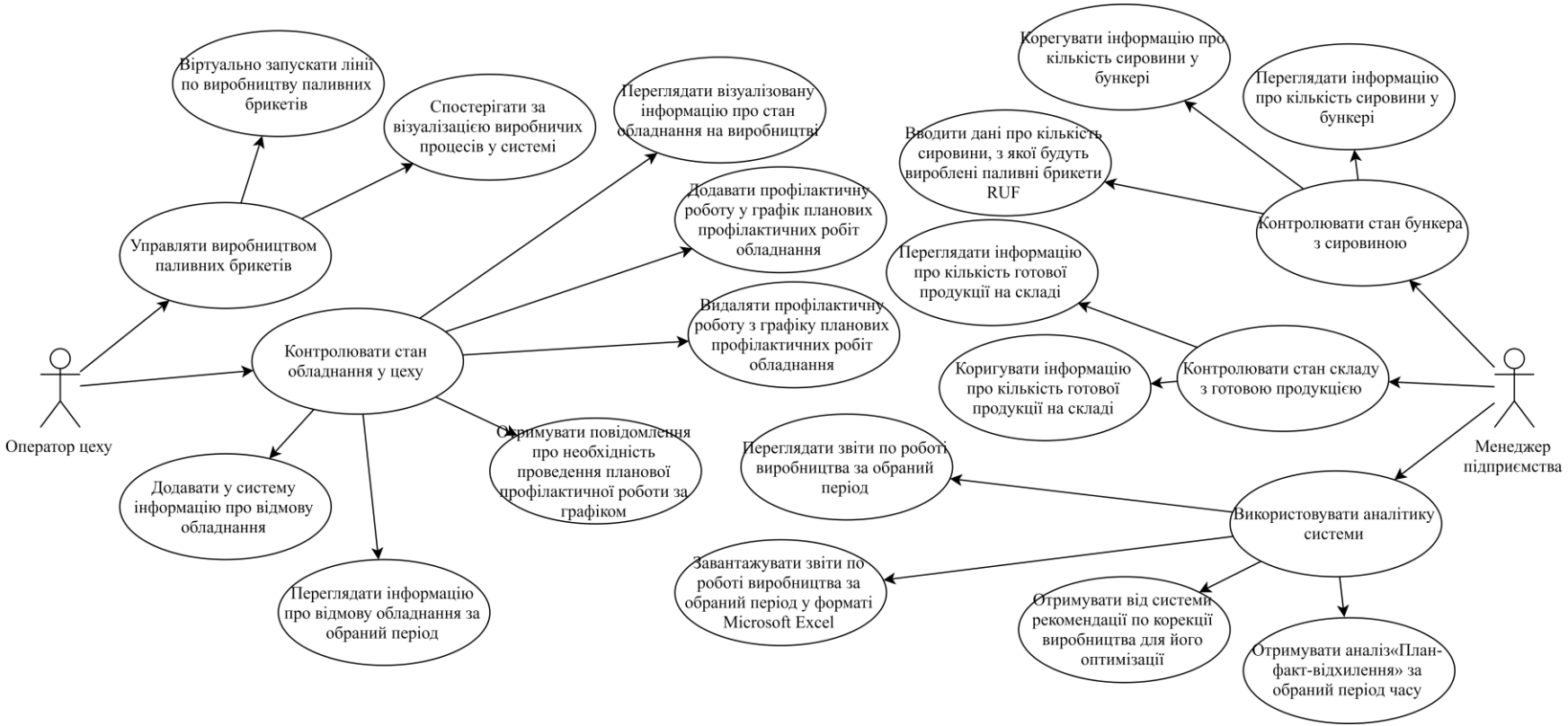
- **«1С: Підприємство»:** програмний продукт компанії «1С», призначений для управління діяльністю на підприємстві. Це потужна ERP система, яка покриває усі області роботи великих підприємств.
- **«Бітрікс24»:** програмний застосунок, який дозволяє керувати роботою підприємства. Охоплює інтеграцію виробництва і операцій, управління трудовими ресурсами, фінансовий менеджмент і управління активами.

Вимоги до програмної системи

Розроблена система має надавати користувачу змогу:

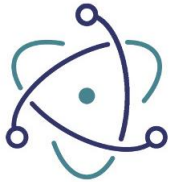
- управляти інформацією про сировину;
- спостерігати за візуалізованими виробничими процесами та віртуально запускати лінії по виробництву паливних брикетів;
- управляти інформацією про кількість готової продукції на складі;
- мати швидкий доступ до звітів по роботі виробництва та аналітики;
- стежити за станом виробничого обладнання.

Діаграма використання системи



Використані технології

React



ELECTRON

Redux

{ REST }

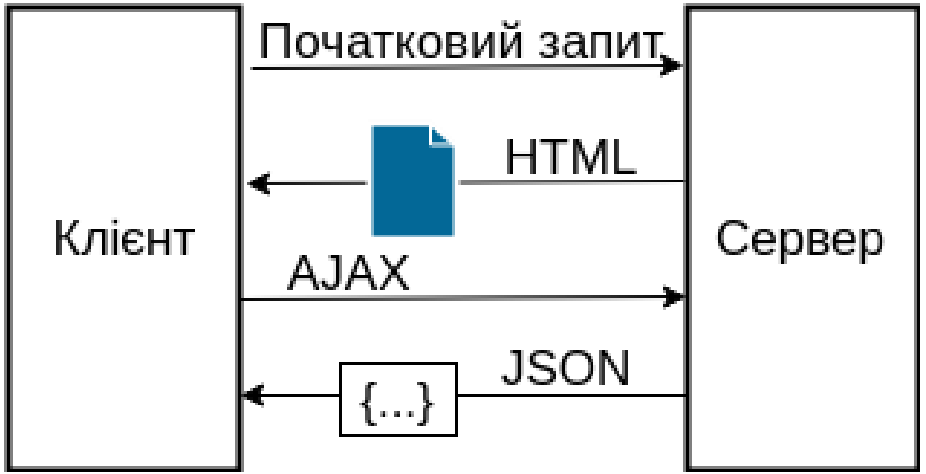
mongoDB

Архітектура розробленого додатку

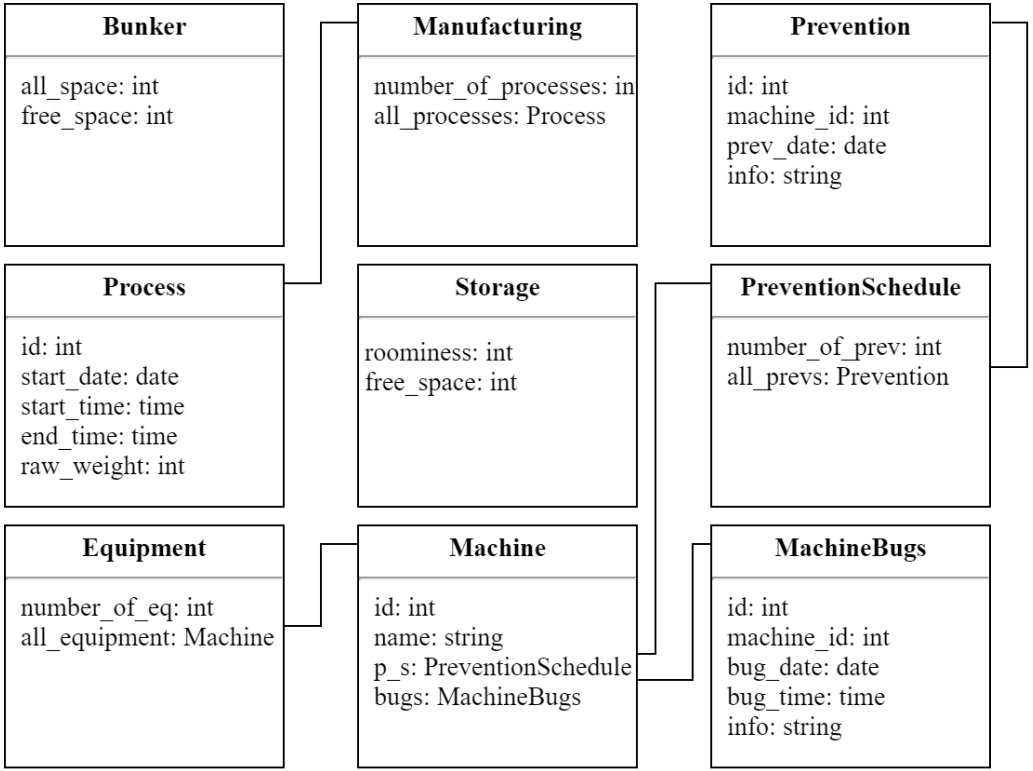
Структура: клієнт-сервер

Стиль взаємодії: REST

Формат обміну даними: JSON



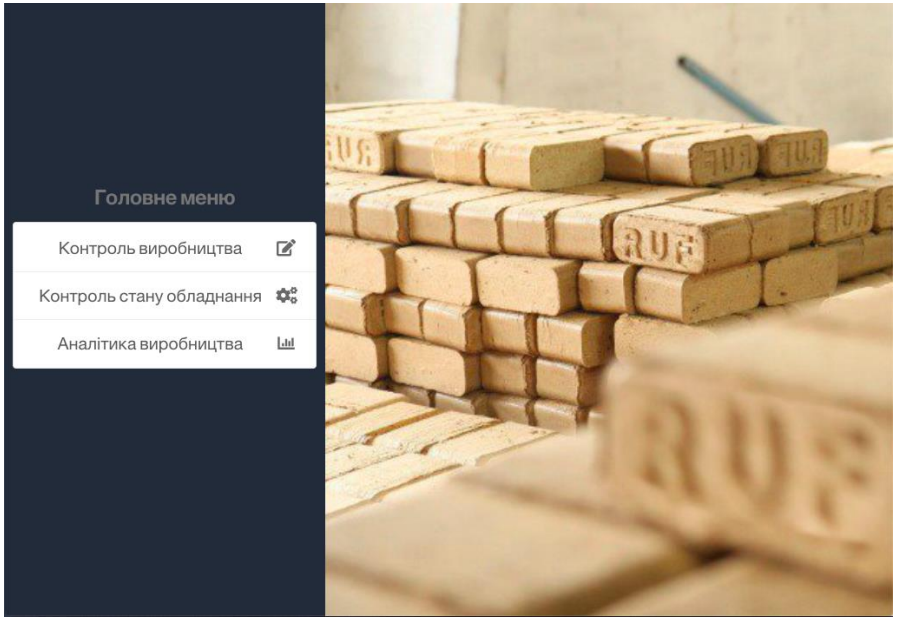
Структура бази даних



Структура системи

Система містить 3 блоки:

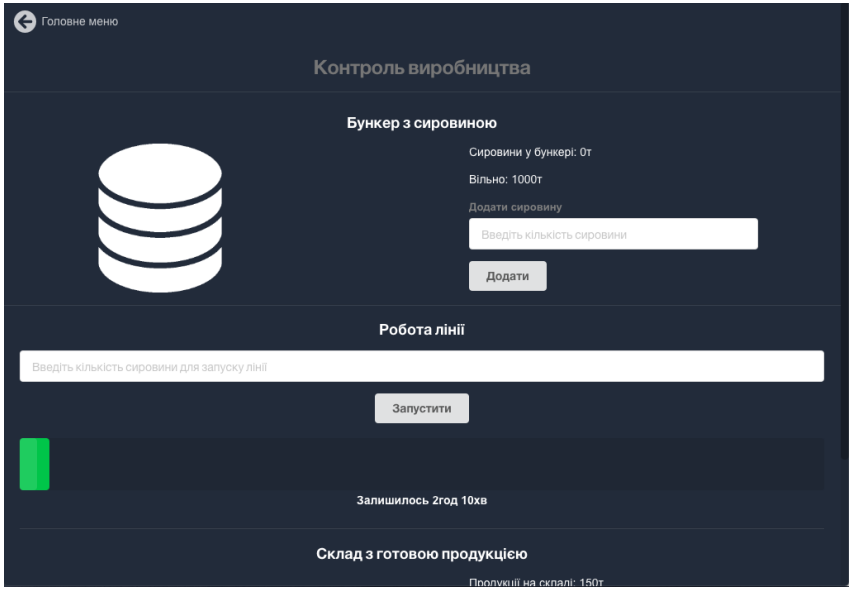
- Контроль виробництва брикетів
- Аналітичний блок
- Контроль стану обладнання

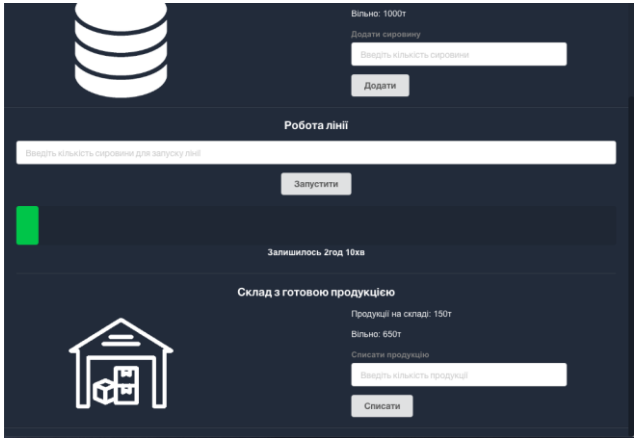


Блок контролю виробництва брикетів

Розділ має 3 умовні блоки:

- Стан бункера з сировиною
- Виробничі процеси
- Стан складу з готовою продукцією

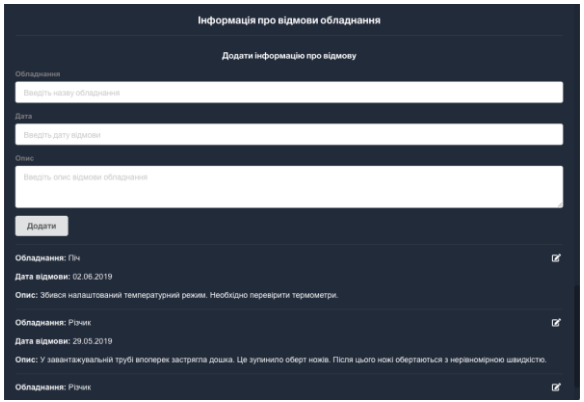




Блок контролю стану обладнання

Розділ має 3 умовні блоки:

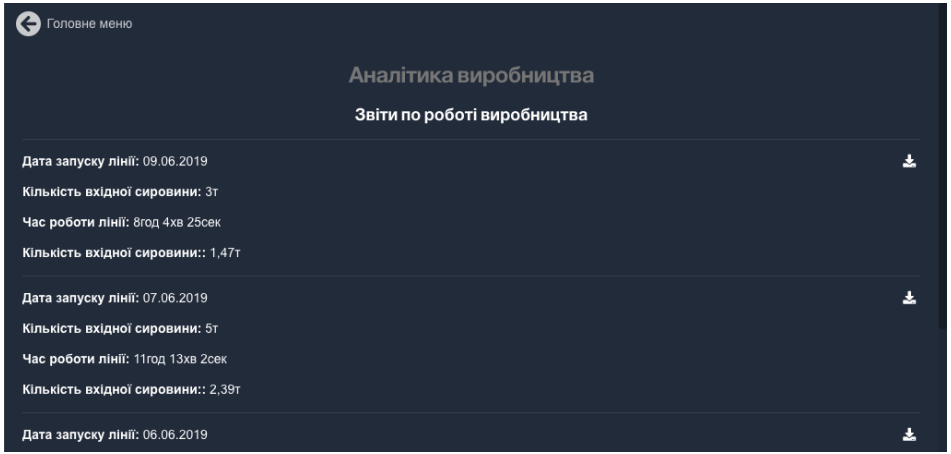
- Інформація про стан обладнання
- Графік профілактичних робіт
- Інформація про відмови обладнання

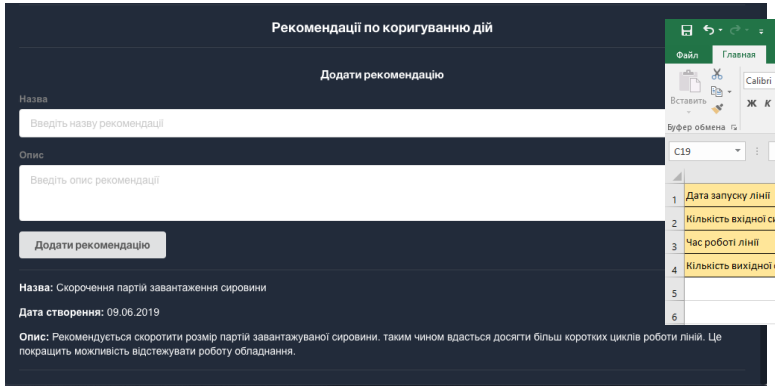


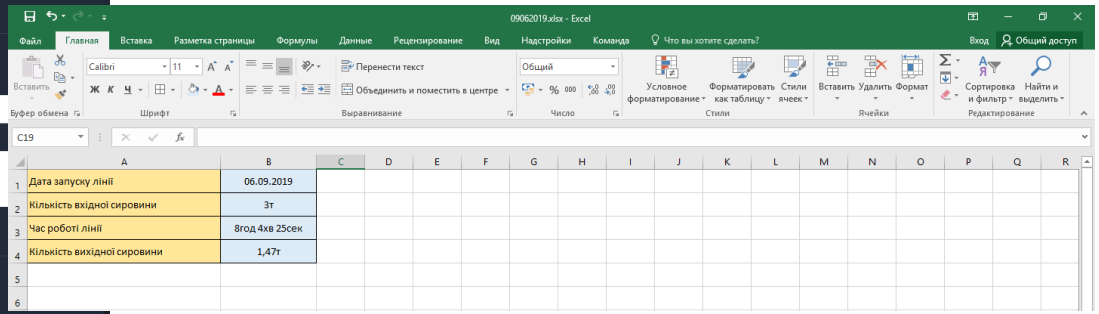
Блок аналітики виробництва

Розділ має 3 умовні блоки:

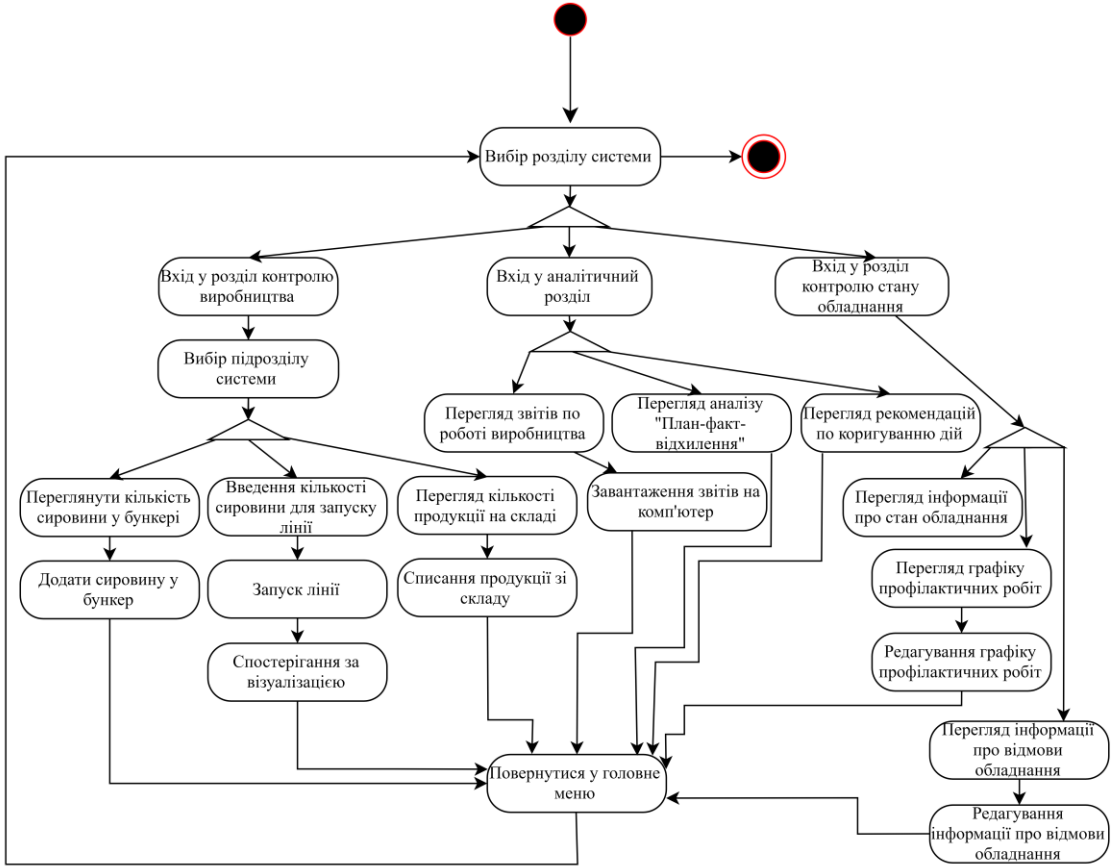
- Звіти по роботі виробництва
- Аналіз «План-факт-відхилення»
- Рекомендовані коригувальні дії







Алгоритм використання системи



Тестування системи

Система була протестована за допомогою методів:

- Manual testing
- Unit-testing

Напрямки подальшого вдосконалення

Система може бути вдосконалена шляхом:

- введення ссортування брикетів по різним параметрам;
- покращення аналітичного блоку;
- підключення декількох виробничих цехів одночасно;
- створення мобільної версії додатку;
- створення браузерної версії додатку;
- зберігання резервних копій даних у хмарному сховищі;
- покращення властивостей інтерфейсу.

Висновки

Під час написання дипломного проекту:

- зібрано дані по роботі підприємства, що виробляє паливних брикетів RUF;
- проаналізовано існуючі програмні рішення;
- проаналізовано та обрано засоби реалізації;
- розроблено програмну систему для обліку та контролю виробництва паливних брикетів RUF;
- протестовано програмну систему;
- створено проектну документацію.

Дякую за увагу!

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ВИРОБНИЦТВА ПАЛИВНИХ БРИКЕТІВ RUF

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Є.О. Пахомова

2018

ЗМІСТ

1. Об’єкт випробувань 3
2. Мета тестування 3
3. Методи тестування 3
4. Засоби та порядок тестування 4

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Програмна система для обліку та контролю виробництва паливних брикетів RUF, яка являє собою реалізований за допомогою web-технологій настільний додаток, створений на основі клієнт-серверної архітектури з використанням фреймворку ReactJS.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок додатку;
- відповідність додатку вимогам продуктивності;
- відповідність додатку обмеженням проектування;
- забезпечення належного рівня надійності додатку;
- зручність роботи з додатком;
- відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється програмний продукт на відповідність функціональним вимогам, а також компоненти програмного коду, які відповідають за отримання даних з зовнішнього сховища.

Використовуються наступні методи:

- функціональне ручне тестування (Manual testing) для перевірки функціональних вимог;
- тестування продуктивності програмного додатку (Stability testing);
- тестування обмежень проектування шляхом введення граничних даних;
- модульне тестування (Unit testing).

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність додатку перевіряється шляхом:

- ручне тестування частини користувача на відповідність функціональним вимогам;
- тестування надійності програмного додатку і цілодобового доступу;
- тестування відповідності статистичних даних, отриманих зі сховища, і даних, які оброблюються програмним додатком;
- тестування продуктивності програмного додатку шляхом вимірювання часу типових запитів та часу отримання відповіді;
- динамічного ручного тестування обмежень проектування шляхом введенням граничних та недопустимих значень в поля, які можна редагувати;
- тестування додатку на різних платформах;
- тестування зручності використання;
- тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ПРОГРАМНА СИСТЕМА ДЛЯ ОБЛІКУ ТА КОНТРОЛЮ ВИРОБНИЦТВА ПАЛИВНИХ БРИКЕТІВ RUF

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Л.А. Люшенко

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Є.О. Пахомова

2019

ЗМІСТ

1. Опис структури системи..... 3
2. Опис блоку контролю виробництва 4
3. Опис блоку контролю стану обладнання..... 6
4. Опис аналітичного блоку 9

1. ОПИС СТРУКТУРИ СИСТЕМИ

Програмний система обліку та контролю виробництва паливних брикетів RUF складається з трьох умовних частин.

Першим, що бачить користувач при вході у систему є головне меню (рис. 1). У ньому містяться такі пункти:

- Контроль виробництва брикетів;
- Аналітичний блок;
- Контроль стану обладнання.

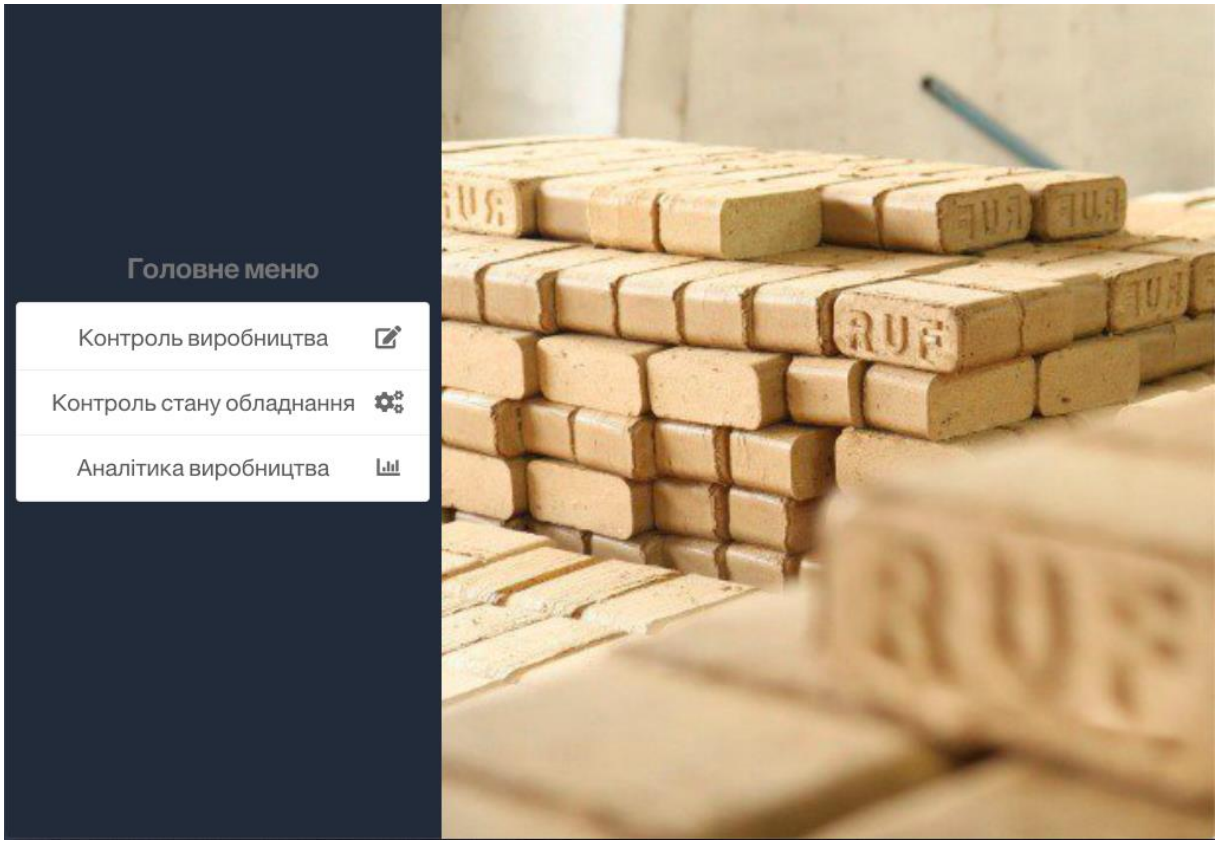


Рис. 1. Головне меню системи

2. ОПИС БЛОКУ КОНТРОЛЮ ВИРОБНИЦТВА

У вікні «*Контроль виробництва*» відображається інформація про виробництво та виробничі процеси, яка ділиться на 3 умовні блоки:

- Стан бункера з сировиною (рис. 2). У цьому блоці міститься інформація про кількість сировини у бункері, а також форма для внесення інформації про додавання сировини у бункер. Списання сировини з бункера буде відбуватися автоматично під час запуску лінії.

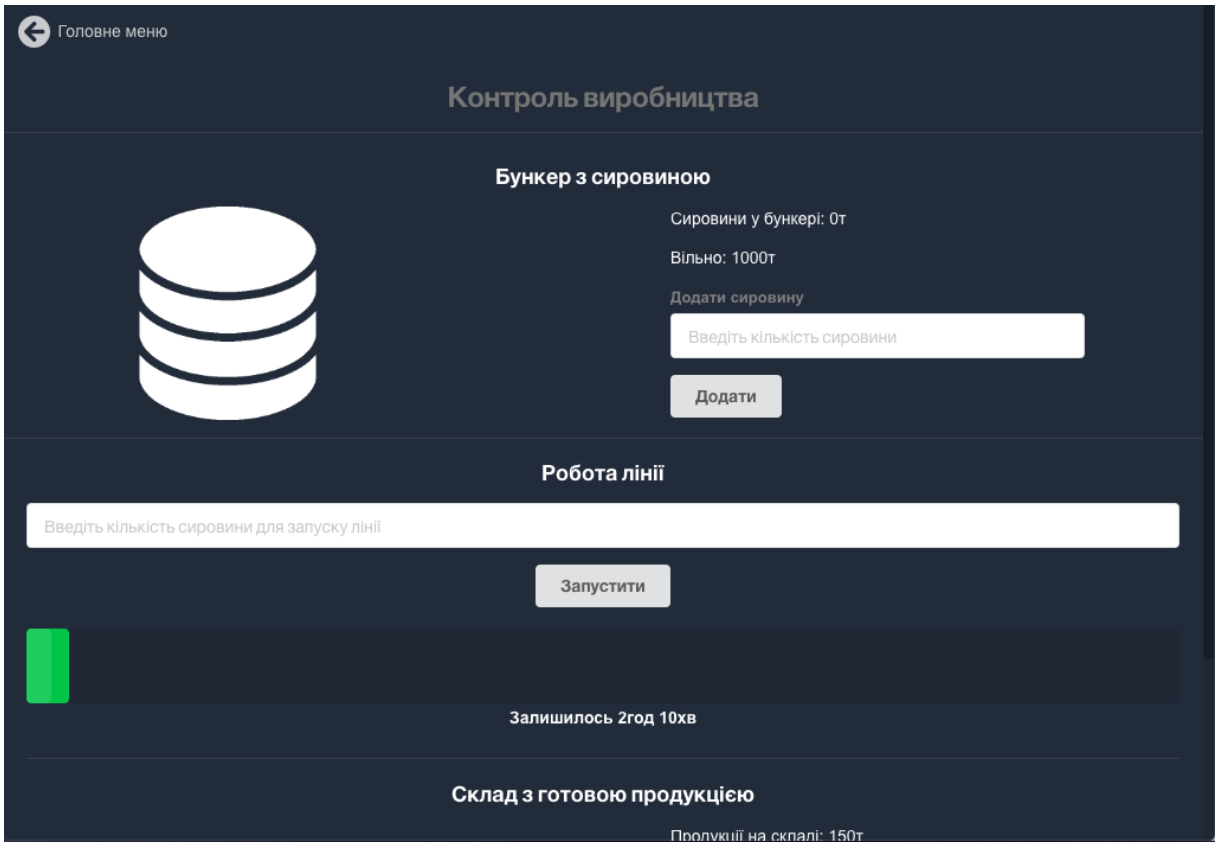


Рис. 2. Стан бункера з сировиною

- Виробничі процеси (рис. 3). У цьому блоці міститься форма для введення даних про кількість сировини, з якої будуть робитися паливні брикети RUF. Також там буде знаходитися кнопка

«Запустити лінію». Ця кнопка буде натискатися оператором одночасно з запуском лінії для візуалізації виробничих процесів. Візуалізація буде відбуватися через індикатори з показниками часу та проробленої роботи у відсотках.

- Стан складу з готовою продукцією (рис. 3). У цьому блоці буде міститися інформація про кількість готової продукції на складі. Інформація буде оновлюватися автоматично після завершення роботі ліній виробництва брикетів. Списання готової продукції зі складу буде проходити через форму, у яку вводяться дані про кількість продукції, яка має бути списана.

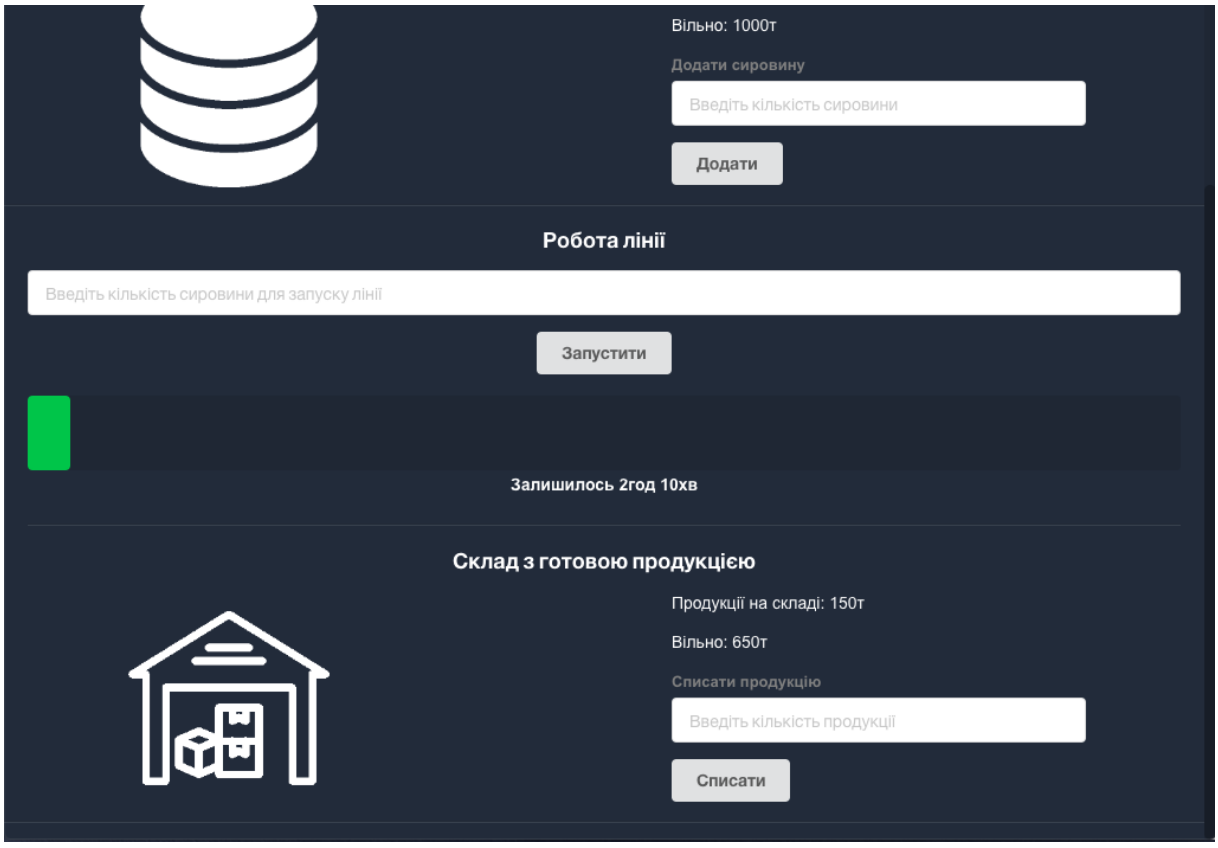


Рис. 3. Виробничі процеси, стан складу

3. ОПИС БЛОКУ КОНТРОЛЮ СТАНУ ОБЛАДНАННЯ

У вікні «*Контроль стану обладнання*» відображається інформація про стан обладнання, яка розділена на 3 умовні блоки:

- Інформація про стан обладнання (рис. 4). У цьому блоці міститиметься інформація з наглядними індикаторами з показниками у відсотках та реальних числах про стан обладнання на виробництві.

Рис. 4. Інформація про стан обладнання

- Графік планових профілактичних робіт (рис. 5). У цьому блоці користувач зможе додати та видалити роботу про профілактику обладнання. Коли час профілактики прийде, система видасть користувачу повідомлення з нагадуванням.

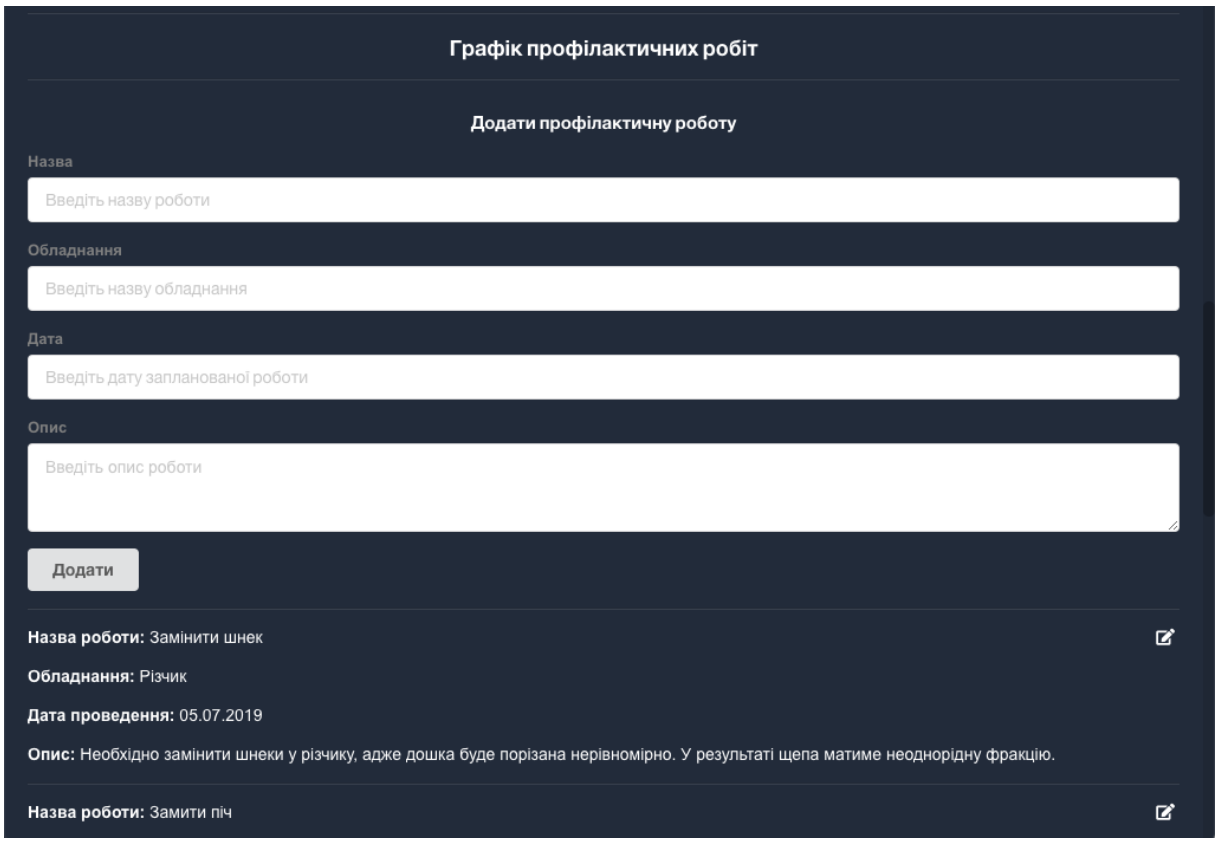


Рис. 5. Графік профілактичних робіт

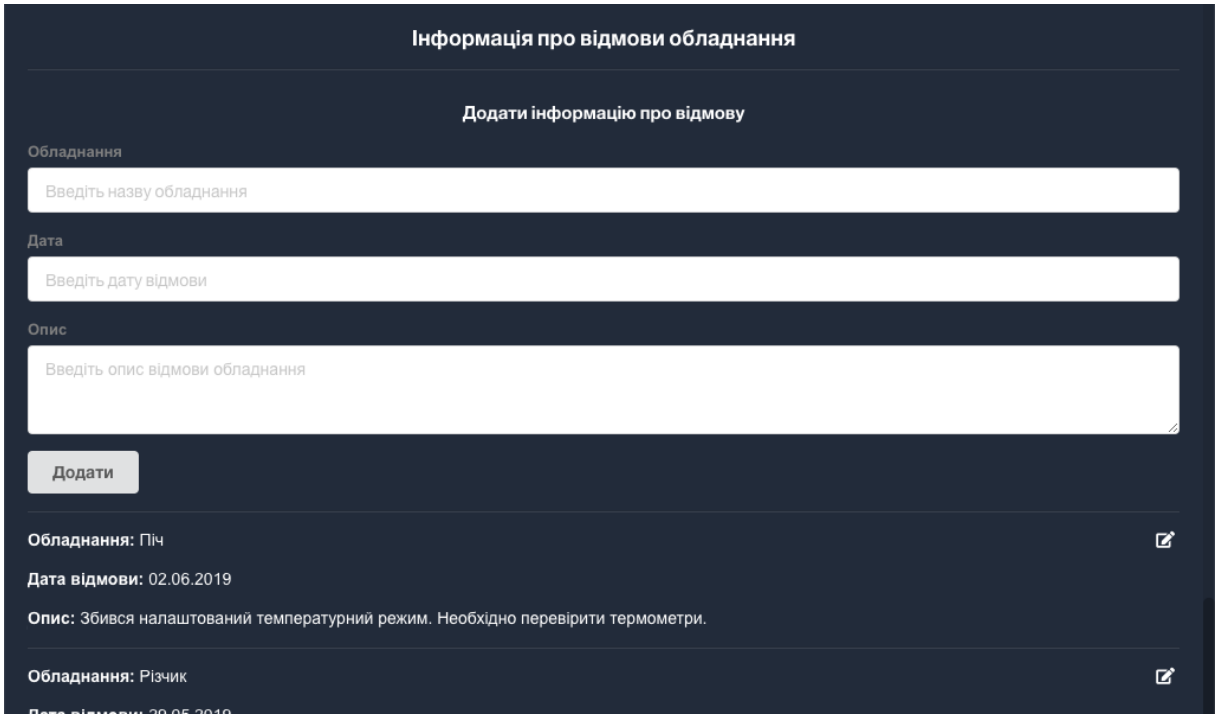


Рис. 6. Інформація про відмови обладнання

- Інформація про відмови обладнання (рис. 6). У цьому блоці можна є спеціальна форма для додавання інформації про відмову обладнання. Інформація про всі попередні відмови може бути виведена відповідно до обраного періоду.

4. ОПИС АНАЛІТИЧНОГО БЛОКУ

У вікні «*Аналітичний блок*» є 3 умовні блоки:

- Звіти по роботі виробництва (рис. 7). У цьому блоці містяться звіти з усіма показниками виробництва. Звіти по обраному періоду можуть бути вигружені з системи у форматі Microsoft Excel для подальшого аналізу сторонніми спеціалістами.

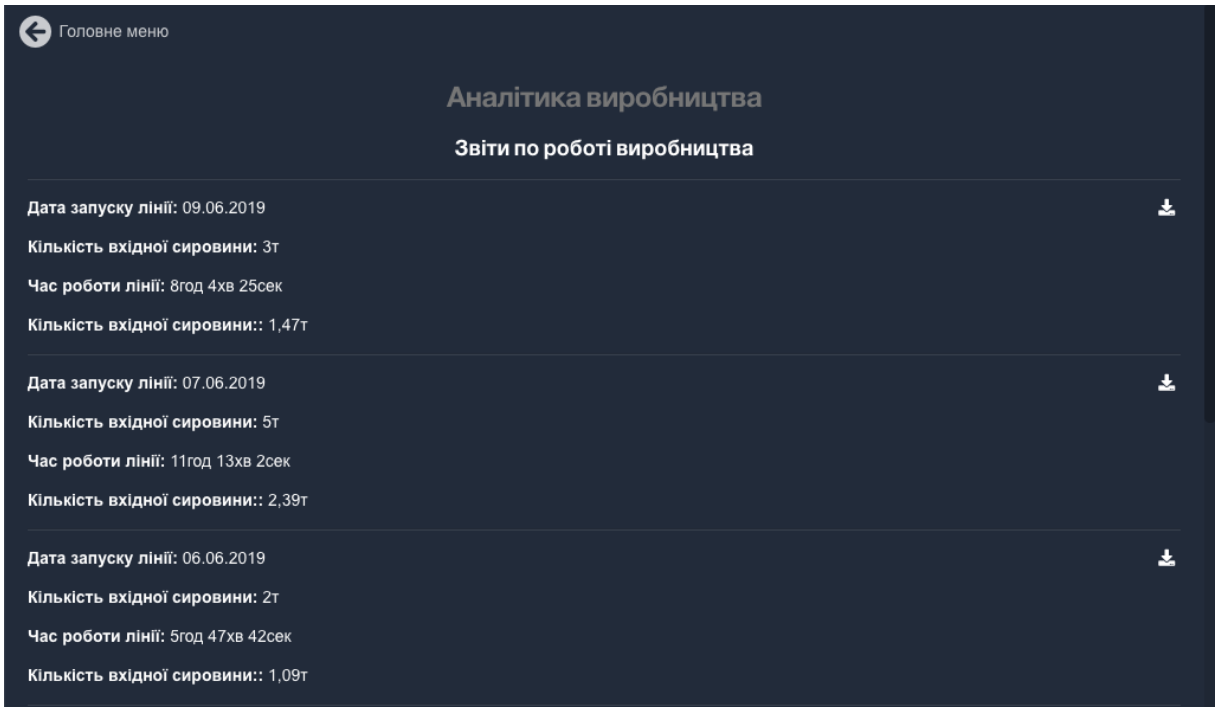


Рис. 7. Звіти по роботі виробництва

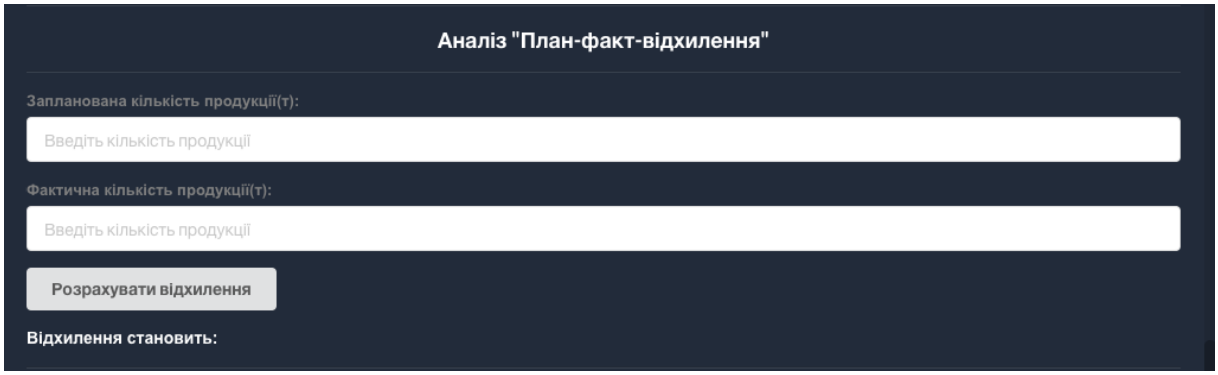


Рис. 8. Аналіз

- Аналіз «План-факт-відхилення» (рис. 8). У цьому блоці можна обрати період для аналізу. Після чого на екран виведуться дані з системи про цей період. Користувач зможе ввести фактичні дані зі складу та побачити відхилення, якщо таке є.
- Рекомендовані коригувальні дії (рис. 9). На основі звітів з системи, користувач має змогу створити рекомендації по корекції виробництва для його оптимізації.

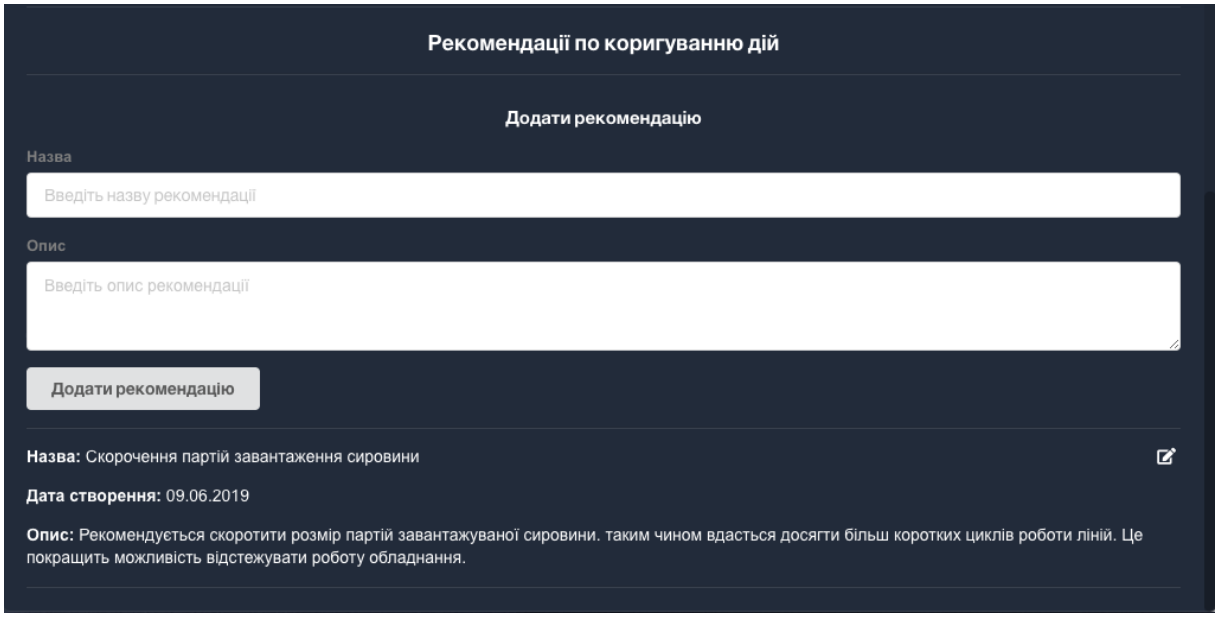


Рис. 9. Рекомендовані коригувальні дії